A Probabilistic Approach to Lexical Semantic Knowledge Acquisition and Structural Disambiguation

Hang LI

A Dissertation
Submitted to the Graduate School of Science
of the University of Tokyo
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Science
in Information Science

July 1998

Abstract

Structural disambiguation in sentence analysis is still a central problem in natural language processing. Past researches have verified that using lexical semantic knowledge can, to a quite large extent, cope with this problem. Although there have been many studies conducted in the past to address the lexical knowledge acquisition problem, further investigation, especially that based on a *principled methodology* is still needed, and this is, in fact, the problem I address in this thesis.

The problem of acquiring and using lexical semantic knowledge, especially that of case frame patterns, can be formalized as follows. A learning module acquires case frame patterns on the basis of some case frame instances extracted from corpus data. A processing (disambiguation) module then refers to the acquired knowledge and judges the degrees of acceptability of some number of new case frames, including *previously unseen* ones.

The approach I adopt has the following characteristics: (1) dividing the problem into three subproblems: case slot generalization, case dependency learning, and word clustering (thesaurus construction). (2) viewing each subproblem as that of statistical estimation and defining probability models for each subproblem, (3) adopting the Minimum Description Length (MDL) principle as learning strategy, (4) employing efficient learning algorithms, and (5) viewing the disambiguation problem as that of statistical prediction.

The need to divide the problem into subproblems is due to the complicatedness of this task, i.e., there are too many relevant factors simply to incorporate all of them into a single model. The use of MDL here leads us to a theoretically sound solution to the 'data sparseness problem,' the main difficulty in a statistical approach to language processing.

In Chapter 3, I define probability models for each subproblem: (1) the hard case slot model and the soft case slot model; (2) the word-based case frame model, the class-based case frame model, and the slot-based case frame model; and (3) the hard co-occurrence model and the soft co-occurrence model. These are respectively the probability models for (1) case slot generalization, (2) case dependency learning, and (3) word clustering. Here the term 'hard' means that the model is characterized by a type of word clustering in which a word can only belong to a single class alone, while 'soft' means that the model is characterized by a type of word clustering in which a word can belong to several different classes.

In Chapter 4, I describe one method for learning the hard case slot model, i.e., generalizing case slots. I restrict the class of hard case slot models to that of tree cut models by using an existing thesaurus. In this way, the problem of generalizing the values of a case slot turns out to be that of estimating a model from the class of tree cut models for some fixed thesaurus tree. I then employ an efficient algorithm, which provably obtains the optimal tree cut model in terms of MDL. This method, in fact, conducts generalization in the following way. When the differences between the frequencies of the nouns in a class are not large enough (relative to the entire data size and the number of the nouns), it generalizes them into the class. When the differences are especially noticeable, on the other hand, it stops generalization at that level.

In Chapter 5, I describe one method for learning the case frame model, i.e., learning dependencies between case slots. I restrict the class of case frame models to that of dependency forest models. Case frame patterns can then be represented as a dependency forest, whose nodes represent case slots and whose directed links represent the dependencies that exist between these case slots. I employ an efficient algorithm to learn the optimal dependency forest model in terms of MDL. This method first calculates a statistic between all node pairs and sorts these node pairs in descending order with respect to the statistic. It then puts a link between the node pair highest in the order, provided that this value is larger than zero. It repeats this process until no node pair is left unprocessed, provided that adding that link will not create a loop in the current dependency graph.

In Chapter 6, I describe one method for learning the hard co-occurrence model, i.e., automatically conducting word clustering. I employ an efficient algorithm to repeatedly estimate a suboptimal MDL model from a class of hard co-occurrence models. The clustering method iteratively merges, for example, noun classes and verb classes in turn, in a bottom up fashion. For each merge it performs, it calculates the decrease in empirical mutual information resulting from merging any noun (or verb) class pair, and performs the merge having the least reduction in mutual information, provided that this reduction in mutual information is less than a threshold, which will vary depending on the data size and the number of classes in the current situation.

In Chapter 7, I propose, for resolving ambiguities, a new method which combines the use of the hard co-occurrence model and that of the tree cut model. In the implementation of this method, the learning module combines with the hard co-occurrence model to cluster words with respect to each case slot, and it combines with the tree cut model for generalizing the values of each case slot by means of a hand-made thesaurus. The disambiguation module first calculates a likelihood value for each interpretation on the basis of hard co-occurrence models and outputs the interpretation with the largest likelihood value; if the likelihood values are equal (most particularly, if all of them are 0), it uses likelihood values calculated on the basis of tree cut models; if the likelihood values are still equal, it makes a default decision.

The accuracy achieved by this method is 85.2%, which is higher than that of state-of-the-art methods.

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Prof. Jun'ichi Tsujii of the University of Tokyo, for his continuous encouragement and guidance. It was Prof. Tsujii who guided me in the fundamentals of natural language processing when I was an undergraduate student at Kyoto University. His many helpful suggestions and comments have also been crucial to the completion of this thesis.

I would also like to express my gratitude to the members of my dissertation committee: Prof. Toshihisa Takagi and Prof. Hiroshi Imai of the University of Tokyo, Prof. Yuji Matsumoto of Nara Institute of Science and Technology (NAIST), and Prof. Kenji Kita of Tokushima University, who have been good enough to give this work a very serious review.

Very special thanks are also due to Prof. Makoto Nagao of Kyoto University for his encouragement and guidance, particularly in his supervision of my master thesis when I was a graduate student at Kyoto University I learned a lot from him, especially in the skills of conducting research. He is one of the persons who continue to influence me strongly in my research carrer, even though the approach I am taking right now is quite different from his own.

I also would like to express my sincere gratitude to Prof. Yuji Matsumoto. He has given me much helpful advice with regard to the conduct of research, both when I was at Kyoto University and after I left there. The use of dependency graphs for representation of case frame patterns was inspired by one of his statements in a personal conversation.

I would like to thank Prof. Jun'ichi Nakamura of Kyoto University, Prof. Satoshi Sato of the Japan Advance Institute of Science and Technology, and other members of the Nagao Lab. for their advice and contributions to our discussions.

The research reported in this dissertation was conducted at C&C Media Research Laboratories, NEC Corporation and the Theory NEC Laboratory, Real World Computing Partnership (RWCP). I would like to express my sincere appreciation to Mr. Katsuhiro Nakamura, Mr. Tomoyuki Fujita, and Dr. Shun Doi of NEC. Without their continuous encouragement and support, I would not have been able to complete this work.

Sincere appreciation also goes to Naoki Abe of NEC. Most of the research reported in this thesis was conducted jointly with him. Without his advice and proposals, I would not have been able to achieve the results represented here. Ideas for the tree cut

model and the hard co-occurrence model came out in discussions with him, and the algorithm 'Find-MDL' was devised on the basis of one of his ideas.

I am deeply appreciative of the encouragement and advice given me by Kenji Yamanishi of NEC, who introduced me to the MDL principle; this was to become the most important stimulus to the idea of conducting this research. He also introduced me to many useful machine learning techniques, that have broadened my outlook toward the field.

I also thank Jun-ichi Takeuchi, Atsuyoshi Nakamura, and Hiroshi Mamitsuka of NEC for their helpful advice and suggestions. Jun-ichi 's introduction to me of the work of Joe Suzuki eventually leads to the development in this study of the case-dependency learning method.

Special thanks are also due to Yuuko Yamaguchi and Takeshi Futagami of NIS who implemented the programs of Find-MDL, 2D-Clustering.

In expressing my appreciation to Yasuharu Den of NAIST, David Carter of Speech Machines, and Takayoshi Ochiai of NIS, I would like them to know how much I had enjoyed the enlightening conversations I had with them.

I am also grateful to Prof. Mark Petersen of Meiji University for what he has taught me about the technical writing of English. Prof. Petersen also helped correct the English of the text in this thesis, and without his help, it would be neither so readable nor so precise.

Yasuharu Den, Kenji Yamanishi, David Carter, and Diana McCarthy of Sussex University read some or all of this thesis and made many helpful comments. Thanks also go to all of them, though the responsibility for flaws and errors it contains remains entirely with me.

I owe a great many thanks to many people who were kind enough to help me over the course of this work. I would like to express here my great appreciation to all of them.

Finally, I also would like to express a deep debt of gratitude to my parents, who instilled in me a love for learning and thinking, and to my wife Hsiao-ya, for her constant encouragement and support.

Contents

Abstract **i**

Acknowledgements **iii**

1 Introduction **1**

1.1 Motivation 1
1.2 Problem Setting 3
1.3 Approach 4
1.4 Organization of the Thesis 7

2 Related Work **9**

2.1 Extraction of Case Frames 9
2.2 Case Slot Generalization 11
2.2.1 Word-based approach and the data sparseness problem 11
2.2.2 Similarity-based approach 13
2.2.3 Class-based approach 14
2.3 Word Clustering 14
2.4 Case Dependency Learning 16
2.5 Structural Disambiguation 16
2.5.1 The lexical approach 16
2.5.2 The combined approach 19
2.6 Word Sense Disambiguation 21
2.7 Introduction to MDL 23
2.7.1 Basics of Information Theory 23
2.7.2 Two-stage code and MDL 26
2.7.3 MDL as data compression criterion 30
2.7.4 MDL as estimation criterion 30
2.7.5 Employing MDL in NLP 34

3 Models for Lexical Knowledge Acquisition **37**

3.1 Case Slot Model 37
3.2 Case Frame Model 40
3.3 Co-occurrence Model 42

3.4 Relations between Models 46
3.5 Discussions 48
3.6 Disambiguation Methods 49
3.7 Summary 52

4 Case Slot Generalization **53**
4.1 Tree Cut Model 53
4.2 MDL as Strategy 56
4.3 Algorithm 59
4.4 Advantages 60
4.5 Experimental Results 62
4.5.1 Experiment 1: qualitative evaluation 62
4.5.2 Experiment 2: pp-attachment disambiguation 65
4.6 Summary 69

5 Case Dependency Learning **73**
5.1 Dependency Forest Model 73
5.2 Algorithm 75
5.3 Experimental Results 77
5.3.1 Experiment 1: slot-based model 77
5.3.2 Experiment 2: slot-based disambiguation 80
5.3.3 Experiment 3: class-based model 81
5.3.4 Experiment 4: simulation 82
5.4 Summary 83

6 Word Clustering **91**
6.1 Parameter Estimation 91
6.2 MDL as Strategy 92
6.3 Algorithm 93
6.4 Experimental Results 96
6.4.1 Experiment 1: qualitative evaluation 96
6.4.2 Experiment 2: compound noun disambiguation 97
6.4.3 Experiment 3: pp-attachment disambiguation 98
6.5 Summary 100

7 Structural Disambiguation **101**
7.1 Procedure 101
7.2 An Analysis System 103
7.3 Experimental Results 104

8 Conclusions **107**
8.1 Summary 107
8.2 Open Problems 107

References **110**

A **127**

A.1 Derivation of Description Length: Two-stage Code 127
A.2 Learning a Soft Case Slot Model 128
A.3 Number of Tree Cuts 129
A.4 Proof of Proposition 1 130
A.5 Equivalent Dependency Tree Models 131
A.6 Proof of Proposition 2 132

Publication List **134**

List of Tables

2.1 Example case frame data. 10
2.2 Example case slot data. 10
2.3 Example case slot data. 11
2.4 Example input data as doubles. 17
2.5 Example input data as triples. 18
2.6 Example input data as quadruples and labels. 18

3.1 Numbers of parameters in case slot models. 39
3.2 Example case frame data generated by a word-based model. 41
3.3 Example case frame data generated by a class-based model. 41
3.4 Example case frame data generated by a slot-based model. 41
3.5 Numbers of parameters in case frame models. 42
3.6 Numbers of parameters in co-occurrence models. 45
3.7 Summary of the formalization. 46

4.1 Number of parameters and KL divergence for the five tree cut models. . 56
4.2 Calculating description length. 58
4.3 Description lengths for the five tree cut models. 58
4.4 Generalization result. 58
4.5 Example input data (for the arg2 slot for 'eat'). 62
4.6 Examples of generalization results. 64
4.7 Required computation time and number of generalized levels. 65
4.8 Number of data items. 66
4.9 PP-attachment disambiguation results. 68
4.10 Example generalization results for SA and MDL. 70
4.11 Some hard examples for LA. 71

5.1 Parameters labeled with each node. 74
5.2 The statistic θ for node pairs. 76
5.3 Verbs appearing most frequently. 77
5.4 Case slots considered. 78
5.5 Verbs and their dependent case slots. 79
5.6 Verbs and their dependent case slots. 87
5.7 Verbs with significant perplexity reduction. 88

5.8 Randomly selected verbs and their perplexities. 88
5.9 PP-attachment disambiguation results. 88

6.1 Compound noun disambiguation results. 98
6.2 PP-attachment disambiguation results. 99
6.3 PP-attachment disambiguation results. 100

7.1 PP-attachment disambiguation results. 105
7.2 Results reported in previous work. 105

8.1 Models proposed. 108
8.2 Algorithm employed. 108

List of Figures

1.1 Organization of this thesis. 7

2.1 Frequency data for the subject slot for verb 'fly.' 11
2.2 Word-based distribution estimated using MLE. 13
2.3 Example co-occurrence data. 15

3.1 An example hard co-occurrence model. 44
3.2 Relations between models. 47

4.1 An example thesaurus. 53
4.2 A tree cut model with [swallow, crow, eagle, bird, bug, bee, insect]. . . . 54
4.3 A tree cut model with [BIRD, bug, bee, insect]. 55
4.4 A tree cut model with [BIRD, INSECT]. 55
4.5 The Find-MDL algorithm. 60
4.6 An example application of Find-MDL. 61
4.7 Example generalization result (for the arg2 slot for 'eat'). 63
4.8 Accuracy-coverage plots for MDL, SA, and LA. 67

5.1 Example dependency forests. 84
5.2 The learning algorithm. 85
5.3 A dependency forest as case frame patterns. 85
5.4 Case frame patterns (dependency forest model) for 'buy.' 86
5.5 Number of links versus data size. 86
5.6 KL divergence versus data size. 89

6.1 A part of a constructed thesaurus. 96
6.2 Accuracy-coverage plots for 2D-Clustering, Brown, and Word-based. . . . 98

7.1 The disambiguation procedure. 102
7.2 Outline of the natural language analysis system. 104

A.1 An example one-way branching binary tree. 130
A.2 Equivalent dependency tree models. 132

Chapter 1

Introduction

... to divide each of the difficulties under examination into as many parts as possible, and as might be necessary for its adequate solution.

- René Descartes

1.1 Motivation

Structural (or syntactic) disambiguation in sentence analysis is still a central problem in natural language processing. To resolve ambiguities completely, we would need to construct a human language 'understanding' system (Johnson-Laird, 1983; Tsujii, 1987; Altmann and Steedman, 1988). The construction of such a system would be extremely difficult, however, if not impossible. For example, when analyzing the sentence

$$\text{I ate ice cream with a spoon,} \tag{1.1}$$

a natural language processing system may obtain two interpretations: "I ate ice cream using a spoon" and "I ate ice cream and a spoon." i.e., a pp-attachment ambiguity may arise, because the prepositional phrase 'with a spoon' can *syntactically* be attached to both 'eat' and 'ice cream.' If a human speaker reads the same sentence, common sense will certainly lead him to assume the former interpretation over the latter, because he understands that: "a spoon is a tool for eating food," "a spoon is not edible," etc. Incorporating such 'world knowledge' into a natural language processing system is highly difficult, however, because of its sheer enormity.

An alternative approach is to make use of only lexical semantic knowledge, specifically case frame patterns (Fillmore, 1968) (or their near equivalents: selectional patterns (Katz and Fodor, 1963), and subcategorization patterns (Pollard and Sag, 1987)). That is, to represent the content of a sentence or a phrase with a 'case frame' having

a 'head'[1] and multiple 'slots,' and to incorporate into a natural language processing system the knowledge of which words can fill into which slot of a case frame.

For example, we can represent the sentence "I ate ice cream" as

(eat (arg1 I) (arg2 ice-cream)),

where the head is 'eat,' the arg1 slot represents the subject and the arg2 slot represents the direct object. The values of the arg1 slot and the arg2 slot are 'I' and 'ice cream,' respectively. Furthermore, we can incorporate as the case frame patterns for the verb 'eat' the knowledge that a member of the word class ⟨animal⟩ can be the value of the arg1 slot and a member of the word class ⟨food⟩ can be the value of the arg2 slot, etc.

The case frames of the two interpretations obtained in the analysis of the above sentence (1.1), then, become

(eat (arg1 I) (arg2 ice-cream) (with spoon))
(eat (arg1 I) (arg2 (ice-cream (with spoon)))).

Referring to the case frame patterns indicating that 'spoon' can be the value of the 'with' slot when the head is 'eat,' and 'spoon' cannot be the value of the 'with' slot when the head is 'ice cream,' a natural language processing system naturally selects the former interpretation and thus resolves the ambiguity.

Previous data analyses have indeed indicated that using lexical semantic knowledge can, to a quite large extent, cope with the structural disambiguation problem (Hobbs and Bear, 1990; Whittemore, Ferrara, and Brunner, 1990). The advantage of the use of lexical knowledge over that of world knowledge is the relative smallness of its amount. By restricting knowledge to that of *relations between words*, the construction of a natural language processing system becomes much easier. (Although the lexical knowledge is still unable to resolve the problem completely, past research suggests that it might be the most realistic path we can take right now.)

As is made clear in the above example, case frame patterns mainly include 'generalized information,' e.g., that a member of the word class ⟨animal⟩ can be the value of the arg2 slot for the verb 'eat.'

Classically, case frame patterns are represented by 'selectional restrictions' (Katz and Fodor, 1963), i.e., discretely represented by semantic features, but it is better to represent them continuously, because a word can be the value of a slot to a certain probabilistic *degree*, as is suggested by the following list (Resnik, 1993b):

(1) Mary drank some wine.
(2) Mary drank some gasoline.
(3) Mary drank some pencils.
(4) Mary drank some sadness.

[1] I slightly abuse terminology here, as 'head' is usually used for subcategorization patterns in the discipline of HPSG, but not in case frame theory.

Furthermore, case frame patterns are not limited to reference to individual case slots. Dependencies between case slots need also be considered. The term 'dependency' here refers to the relationship that may exist between case slots and that indicates strong co-occurrence between the values of those case slots. For example, consider the following sentences:[2]

(1) She flies jets.
(2) That airline company flies jets.
(3) She flies Japan Airlines.
(4) *That airline company flies Japan Airlines.

(1.2)

We see that an 'airline company' can be the value of the arg1 slot, when the value of the arg2 slot is an 'airplane' but not when it is an 'airline company.' These sentences indicate that the possible values of case slots depend in general on those of others: dependencies between case slots exist.[3]

Another consensus on lexical semantic knowledge in recent studies is that it is preferable to learn lexical knowledge automatically from corpus data. Automatic acquisition of lexical knowledge has the merits of (1) saving the cost of defining knowledge by hand, (2) doing away with the subjectivity inherent in human-defined knowledge, and (3) making it easier to adapt a natural language processing system to a new domain.

Although there have been many studies conducted in the past (described here in Chapter 2) to address the lexical knowledge acquisition problem, further investigation, especially that based on a *principled methodology* is still needed, and this is, in fact, the problem I address in this thesis.

The search for a mathematical formalism for lexical knowledge acquisition is not only motivated by concern for logical niceties; I believe that it can help to better cope with practical problems (for example, the disambiguation problem). The ultimate outcome of the investigations in this thesis, therefore, should be a formalism of lexical knowledge acquisition and at the same time a high-performance disambiguation method.

1.2 Problem Setting

The problem of acquiring and using lexical semantic knowledge, especially that of case frame patterns, can be formalized as follows. A learning module acquires case frame patterns on the basis of some case frame instances extracted from corpus data. A processing (disambiguation) module then refers to the acquired knowledge and judges

[2] '*' indicates an unacceptable natural language expression.

[3] One may argue that 'fly' has different word senses in these sentences and for each of these word senses there is no dependency between the case slots. Word senses are in general difficult to define precisely, however. I think that it is preferable not to resolve them until doing so is necessary in a particular application. That is to say that, in general, case dependencies do exist and the development of a method for learning them is needed.

the degrees of acceptability of some new case frames, including *previously unseen* ones. The goals of learning are to represent more *compactly* the given case frames, and to judge more *correctly* the degrees of acceptability of new case frames.

In this thesis, I propose a probabilistic approach to lexical knowledge acquisition and structural disambiguation.

1.3 Approach

In general, a machine learning process consists of three elements: model, strategy (criterion), and algorithm. That is, when we conduct machine learning, we need consider (1) what kind of model we are to use to represent the problem, (2) what kind of strategy we should adopt to control the learning process, and (3) what kind of algorithm we should employ to perform the learning task. We need to consider each of these elements here.

Division into subproblems

The lexical semantic knowledge acquisition problem is a quite complicated task, and there are too many relevant factors (generalization of case slot values, dependencies between case slots, etc.) to simply incorporate all of them into a single model. As a first step, I divide the problem into three subproblems: case slot generalization, case dependency learning, and word clustering (thesaurus construction).

I define probability models (probability distributions) for each subproblem and view the learning task of each subproblem as that of estimating its corresponding probability models based on corpus data.

Probability models

We can assume that case slot data for a case slot for a verb are generated on the basis of a conditional probability distribution that specifies the conditional probability of a noun given the verb and the case slot. I call such a distribution a 'case slot model.' When the conditional probability of a noun is defined as the conditional probability of the noun class to which the noun belongs, divided by the size of the noun class, I call the case slot model a 'hard case slot model.' When the case slot model is defined as a finite mixture model, namely a linear combination of the word probability distributions within individual noun classes, I call it a 'soft case slot model.'

Here the term 'hard' means that the model is characterized by a type of word clustering in which a word can only belong to a single class alone, while 'soft' means that the model is characterized by a type of word clustering in which a word can belong to several different classes.

I formalize the problem of generalizing the values of a case slot as that of estimating a hard (or soft) case slot model. The generalization problem, then, turns out to be

that of selecting a model, from a class of hard (or soft) case slot models, which is most likely to have given rise to the case slot data.

We can assume that case frame data for a verb are generated according to a multi-dimensional joint probability distribution over random variables that represent the case slots. I call the distribution a 'case frame model.' I further classify this case frame model into three types of probability models each reflecting the type of its random variables: the 'word-based case frame model,' the 'class-based case frame model,' and the 'slot-based case frame model.'

I formalize the problem of learning dependencies between case slots as that of estimating a case frame model. The dependencies between case slots are represented as probabilistic dependencies between random variables.

We can assume that co-occurrence data for nouns and verbs with respect to a slot are generated based on a joint probability distribution that specifies the co-occurrence probabilities of noun verb pairs. I call such a distribution a 'co-occurrence model.' I call this co-occurrence model a 'hard co-occurrence model,' when the joint probability of a noun verb pair is defined as the product of the following three elements: (1) the joint probability of the noun class and the verb class to which the noun and the verb respectively belong, (2) the conditional probability of the noun given its noun class, and (3) the conditional probability of the verb given its verb class. When the co-occurrence model is defined as a double mixture model, namely, a double linear combination of the word probability distributions within individual noun classes and those within individual verb classes, I call it a 'soft co-occurrence model.'

I formalize the problem of clustering words as that of estimating a hard (or soft) co-occurrence model. The clustering problem, then, turns out to be that of selecting a model from a class of hard (or soft) co-occurrence models, which is most likely to have given rise to the co-occurrence data.

MDL as strategy

For all subproblems, the learning task turns out to be that of selecting the best model from among a class of models. The question now is what the learning strategy (or criterion) is to be. I employ here the Minimum Description Length (MDL) principle. The MDL principle is a principle for both data compression and statistical estimation (described in Chapter 2).

MDL provides a theoretically way to deal with the 'data sparseness problem,' the main difficulty in a statistical approach to language processing. At the same time, MDL leads us to an information-theoretic solution to the lexical knowledge acquisition problem, in which case frames are viewed as *structured data*, and the learning process turns out to be that of *data compression*.

Efficient algorithms

In general, there is a trade-off between model classes and algorithms. A complicated model class would be precise enough for representing a problem, but it might be difficult to learn in terms of learning accuracy and computation time. In contrast, a simple model class might be easy to learn, but it would be too simplistic for representing a problem.

In this thesis, I place emphasis on efficiency and restrict a model class when doing so is still reasonable for representing the problem at hand.

For the case slot generalization problem, I make use of an existing thesaurus and restrict the class of hard case slot models to that of 'tree cut models.' I also employ an efficient algorithm, which provably obtains the optimal tree cut model in terms of MDL.

For the case dependency learning problem, I restrict the class of case frame models to that of 'dependency forest models,' and employ another efficient algorithm to learn the optimal dependency forest model in terms of MDL.

For the word clustering problem, I address the issue of estimating the hard co-occurrence model, and employ an efficient algorithm to repeatedly estimate a suboptimal MDL model from a class of hard co-occurrence models.

Disambiguation methods

I then view the structural disambiguation problem as that of *statistical prediction*. Specifically, the likelihood value of each interpretation (case frame) is calculated on the basis of the above models, and the interpretation with the largest likelihood value is output as the analysis result.

I have devised several disambiguation methods along this line.

One of them is especially useful when the data size for training is at the level of that currently available. In implementation of this method, the learning module combines with the hard co-occurrence model to cluster words with respect to each case slot, and it combines with the tree cut model to generalize the values of each case slot by means of a hand-made thesaurus. The disambiguation module first calculates a likelihood value for each interpretation on the basis of hard co-occurrence models and outputs the interpretation with the largest likelihood value; if the likelihood values are equal (most particularly, if all of them are 0), it uses likelihood values calculated on the basis of tree cut models; if the likelihood values are still equal, it makes a default decision.

The accuracy achieved by this method is 85.2%, which is higher than that of state-of-the-art methods.

1.4 Organization of the Thesis

This thesis is organized as follows. In Chapter 2, I review previous work on lexical semantic knowledge acquisition and structural disambiguation. I also introduce the MDL principle. In Chapter 3, I define probability models for each subproblem of lexical semantic knowledge acquisition. In Chapter 4, I describe the method of using the tree cut model to generalize case slots. In Chapter 5, I describe the method of using the dependency forest model to learn dependencies between case slots. In Chapter 6, I describe the method of using the hard co-occurrence model to conduct word clustering. In Chapter 7, I describe the practical disambiguation method. In Chapter 8, I conclude the thesis with some remarks (see Figure 1.1).

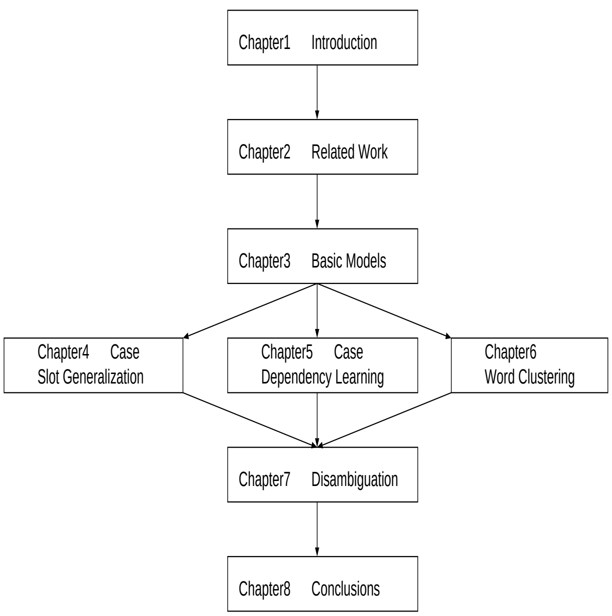


Figure 1.1: Organization of this thesis.

Chapter 2

Related Work

Continue to cherish old knowledge so as to continue to discover new.

- Confucius

In this chapter, I review previous work on lexical knowledge acquisition and disambiguation. I also introduce the MDL principle.

2.1 Extraction of Case Frames

Extracting case frame instances *automatically* from corpus data is a difficult task, because when conducting extraction, ambiguities may arise, and we need to exploit lexical semantic knowledge to resolve them. Since our goal of extraction is indeed to acquire such knowledge, we are faced with the problem of which is to come first, the chicken or the egg.

Although there have been many methods proposed to automatically extract case frames from corpus data, their accuracies do not seem completely satisfactory, and the problem still needs investigation.

Manning (1992), for example, proposes extracting case frames by using a finite state parser. His method first uses a statistical tagger (cf., (Church, 1988; Kupiec, 1992; Charniak et al., 1993; Merialdo, 1994; Nagata, 1994; Schütze and Singer, 1994; Brill, 1995; Samuelsson, 1995; Ratnaparkhi, 1996; Haruno and Matsumoto, 1997)) to assign a part of speech to each word in the sentences of a corpus. It then uses the finite state parser to parse the sentences and note case frames following verbs. Finally, it filters out statistically unreliable extracted results on the basis of hypothesis testing (see also (Brent, 1991; Brent, 1993; Smadja, 1993; Chen and Chen, 1994; Grefenstette, 1994)).

Briscoe and Carroll (1997) extracted case frames by using a probabilistic LR parser. This parser first parses sentences to obtain analyses with 'shallow' phrase structures, and assigns a likelihood value to each analysis. An extractor then extracts case frames

from the most likely analyses (see also (Hindle and Rooth, 1991; Grishman and Sterling, 1992)).

Utsuro, Matsumoto, and Nagao (1992) propose extracting case frames from a parallel corpus in two different languages. Exploiting the fact that a syntactic ambiguity found in one language may not exist at all in another language, they conduct pattern matching between case frames of translation pairs given in the corpus and choose the best matched case frames as extraction results (see also (Matsumoto, Ishimoto, and Utsuro, 1993)).

An alternative to the automatic approach is to employ a semi-automatic method, which can provide much more reliable results. The disadvantage, however, is its requirement of having disambiguation decisions made by a human, and how to reduce the cost of human intervention becomes an important issue.

Carter (1997) developed an interaction system for effectively collecting case frames semi-automatically. This system first presents a user with a range of properties that may help resolve ambiguities in a sentence. The user then designates the value of one of the properties, the system discards those interpretations which are inconsistent with the designation, and it re-displays only the properties which remain. After several such interactions, the system obtains a most likely correct case frame of a sentence (see also (Marcus, Santorini, and Marcinkiewicz, 1993)).

Using any one of the methods, we can extract case frame instances for a verb, to obtain data like that shown in Table 2.1, although no method guarantees that the extracted results are completely correct. In this thesis, I refer to this type of data as 'case frame data.' If we restrict our attention on a specific slot, we obtain data like that shown in Table 2.2. I refer to this type of data as 'case slot data.'

Table 2.1: Example case frame data.

(fly (arg1 girl)(arg2 jet))
(fly (arg1 company)(arg2 jet))
(fly (arg1 girl)(arg2 company))

Table 2.2: Example case slot data.

Verb	Slot name	Slot value
fly	arg1	girl
fly	arg1	company
fly	arg1	girl

2.2 Case Slot Generalization

One case-frame-pattern acquisition problem is that of generalization of (values of) case slots; this has been intensively investigated in the past.

2.2.1 Word-based approach and the data sparseness problem

Table 2.3 shows some example cast slot data for the arg1 slot for the verb 'fly.' By counting occurrences of each noun at the slot, we can obtain frequency data shown in Figure 2.1.

Table 2.3: Example case slot data.

Verb	Slot name	Slot value
fly	arg1	bee
fly	arg1	bird
fly	arg1	bird
fly	arg1	crow
fly	arg1	bird
fly	arg1	eagle
fly	arg1	bee
fly	arg1	eagle
fly	arg1	bird
fly	arg1	crow

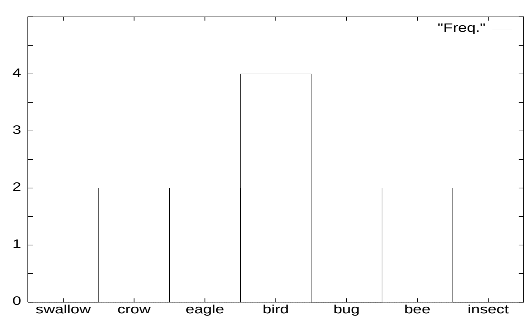


Figure 2.1: Frequency data for the subject slot for verb 'fly.'

The problem of learning 'case slot patterns' for a slot for a verb can be viewed as the problem of estimating the underlying conditional probability distribution which

gives rise to the corresponding case slot data. The conditional distribution is defined as

$$P(n|v,r), \tag{2.1}$$

where random variable n represents a value in the set of nouns $\mathcal{N} = \{n_1, n_2, \cdots, n_N\}$, random variable v a value in the set of verbs $\mathcal{V} = \{v_1, v_2, \cdots, v_V\}$, and random variable r a value in the set of slot names $\mathcal{R} = \{r_1, r_2, \cdots, r_R\}$. Since random variables take on words as their values, this type of probability distribution is often referred to as a 'word-based model.' The degree of noun n's being the value of slot r for verb v[1] is represented by a conditional probability.

Another way of learning case slot patterns for a slot for a verb is to calculate the 'association ratio' measure, as proposed in (Church et al., 1989; Church and Hanks, 1989; Church et al., 1991). The association ratio is defined as

$$S(n|v,r) = \log \frac{P(n|v,r)}{P(n)}, \tag{2.2}$$

where n assumes a value from the set of nouns, v from the set of verbs and r from the set of slot names. The degree of noun n being the value of slot r for verb v is represented as the ratio between a conditional probability and a marginal probability.

The two measures in fact represent two different aspects of case slot patterns. The former indicates the *relative frequency* of a noun's being the slot value, while the latter indicates the *strength of associativeness* between a noun and the verb with respect to the slot. The advantage of the latter may be that it takes into account of the influence of the marginal probability $P(n)$ on the conditional probability $P(n|v,r)$. The advantage of the former may be its ease of use in disambiguation as a likelihood value.

Both the use of the conditional probability and that of the association ratio may suffer from the 'data sparseness problem,' i.e., the number of parameters in the conditional distribution defined in (2.1) is very large, and accurately estimating them is difficult with the amount of data typically available.

When we employ Maximum Likelihood Estimation (MLE) to estimate the parameters, i.e., when we estimate the conditional probability $P(n|v,r)$ as[2]

$$\hat{P}(n|v,r) = \frac{f(n|v,r)}{f(v,r)},$$

where $f(n|v,r)$ stands for the frequency of noun n being the value of slot r for verb v, $f(v,r)$ the total frequency of r for v (Figure 2.2 shows the results for the data in Figure 2.1), we may obtain quite poor results. Most of the probabilities might be estimated as 0, for example, just because a possible value of the slot in question happens not to appear.

[1] Hereafter, I will sometimes use the same symbol to denote both a random variable and one of its values; it should be clear from the context, which it is denoting at any given time.

[2] Throughout this thesis, $\hat{\theta}$ denotes an estimator (or an estimate) of θ.

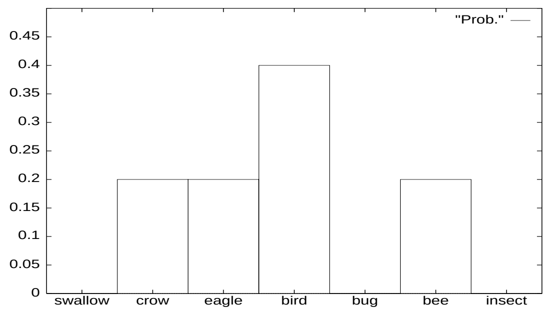


Figure 2.2: Word-based distribution estimated using MLE.

To overcome this problem, we can smooth the probabilities by resorting to statistical techniques (Jelinek and Mercer, 1980; Katz, 1987; Gale and Church, 1990; Ristad and Thomas, 1995). We can, for example, employ an extended version of the Laplace's Law of Succession (cf., (Jeffreys, 1961; Krichevskii and Trofimov, 1981)) to estimate $P(n|v,r)$ as

$$\hat{P}(n|v,r) = \frac{f(n|v,r) + 0.5}{f(v,r) + 0.5 \cdot N}$$

where N denotes the size of the set of nouns.[3]

The results may still not be satisfactory, however. One possible way to cope better with the data sparseness problem is to exploit *additional* knowledge or data rather than make use of only related case slot data. Two such approaches have been proposed previously: one is called the 'similarity-based approach,' the other the 'class-based approach.'

2.2.2 Similarity-based approach

Grishman and Sterling (1994) propose to estimate conditional probabilities by using other conditional probabilities under contexts of similar words, where the similar words themselves are collected on the basis of corpus data. Their method estimates the conditional probability $P(n|v,r)$ as

$$\hat{P}(n|v,r) = \sum_{v'} \lambda(v,v') \cdot \hat{P}(n|v',r),$$

where v' represents a verb similar to verb v, and $\lambda(v,v')$ the similarity between v and v'. That is, it smoothes a conditional probability by taking the weighted average of

[3]This smoothing method can be justified from the viewpoint of Bayesian Estimation. The estimate is in fact the Bayesian estimate with Jeffrey's Prior being the prior probability.

other conditional probabilities under contexts of similar words using similarities as the weights. Note that the equation

$$\sum_{v'} \lambda(v, v') = 1$$

must hold. The advantage of this approach is that it relies only on corpus data. (Cf., (Dagan, Marcus, and Makovitch, 1992; Dagan, Pereira, and Lee, 1994; Dagan, Pereira, and Lee, 1997).)

2.2.3 Class-based approach

A number of researchers have proposed to employ 'class-based models,' which use classes of words rather than individual words.

An example of a class-based approach is Resnik's method of learning case slot patterns by calculating the 'selectional association' measure (Resnik, 1993a; Resnik, 1993b). The selectional association is defined as:

$$A(n|v,r) = \max_{C \ni n} \left(P(C|v,r) \cdot \log \frac{P(C|v,r)}{P(C)} \right), \tag{2.3}$$

where n represents a value in the set of nouns, v a value in the set of verbs and r a value in the set of slot names, and C denotes a class of nouns present in a given thesaurus. (See also (Framis, 1994; Ribas, 1995).) This measure, however, is based on heuristics, and thus can be difficult to justify theoretically.

Other class-based methods for case slot generalization are also proposed (Almuallim et al., 1994; Tanaka, 1994; Tanaka, 1996; Utsuro and Matsumoto, 1997; Miyata, Utsuro, and Matsumoto, 1997).

2.3 Word Clustering

Automatically clustering words or constructing a thesaurus can also be considered to be a class-based approach, and it helps cope with the data sparseness problem not only in case frame pattern acquisition but also in other natural language learning tasks.

If we focus our attention on one case slot, we can obtain 'co-occurrence data' for verbs and nouns with respect to that slot. Figure 2.3, for example, shows such data, in this case, counts of co-occurrences of verbs and their arg2 slot values (direct objects). We can classify words by using such co-occurrence data on the assumption that semantically similar words have similar co-occurrence patterns.

A number of methods have been proposed for clustering words on the basis of co-occurrence data. Brown et al. (1992), for example, propose a method of clustering words on the basis of MLE in the context of n-gram estimation. They first define an n-gram class model as

$$P(w_n|w_1^{n-1}) = P(w_n|C_n) \cdot P(C_n|C_1^{n-1}),$$

	eat	drink	make
wine	0	3	1
beer	0	5	1
bread	4	0	2
rice	4	0	0

Figure 2.3: Example co-occurrence data.

where C represents a word class. They then view the clustering problem as that of partitioning the vocabulary (a set of words) into a designated number of word classes whose resulting 2-gram class model has the maximum likelihood value with respect to a given word sequence (i.e., co-occurrence data). Brown et al have also devised an efficient algorithm for performing this task, which turns out to iteratively merge the word class pair having the least reduction in empirical mutual information until the number of classes created equals the designated number. The disadvantage of this method is that one has to designate in advance the number of classes to be created, with no guarantee at all that this number will be optimal.

Pereira, Tishby, and Lee (1993) propose a method of clustering words based on co-occurrence data over two sets of words. Without loss of generality, suppose that the two sets are a set of nouns $\mathcal{N}$ and a set of verbs $\mathcal{V}$, and that a sample of co-occurrence data is given as $(n_i, v_i), n_i \in \mathcal{N}, v_i \in \mathcal{V}, i = 1, \cdots, s$. They define

$$P(n, v) = \sum_C P(C) \cdot P(n|C) \cdot P(v|C)$$

as a model which can give rise to the co-occurrence data, where C represents a class of nouns. They then view the problem of clustering nouns as that of estimating such a model. The classes obtained in this way, which they call 'soft clustering,' have the following properties: (1) a noun can belong to several different classes, and (2) each class is characterized by a membership distribution. They devised an efficient clustering algorithm based on 'deterministic annealing technique.'[4] Conducting soft clustering makes it possible to cope with structural and word sense ambiguity at the

[4]Deterministic annealing is a computation technique for finding the global optimum (minimum) value of a cost function (Rose, Gurewitz, and Fox, 1990; Ueda and Nakano, 1998). The basic idea is to conduct minimization by using a number of 'free energy' functions parameterized by 'temperatures' for which free energy functions with high temperatures *loosely* approximate a target function,

same time, but it also requires more training data and makes the learning process more computationally demanding.

Tokunaga, Iwayama, and Tanaka (1995) point out that, for disambiguation purposes, it is necessary to construct one thesaurus for each case slot on the basis of co-occurrence data concerning to that slot. Their experimental results indicate that, for disambiguation, the use of thesauruses constructed from data specific to the target slot is preferable to the use of thesauruses constructed from data non-specific to the slot.

Other methods for automatic word clustering have also been proposed (Hindle, 1990; Pereira and Tishby, 1992; McKeown and Hatzivassiloglou, 1993; Grefenstette, 1994; Stolcke and Omohundro, 1994; Abe, Li, and Nakamura, 1995; McMahon and Smith, 1996; Ushioda, 1996; Hogenhout and Matsumoto, 1997).

2.4 Case Dependency Learning

There has been no method proposed to date, however, that learns dependencies between case slots. In past research, methods of resolving ambiguities have been based, for example, on the assumption that case slots are mutually independent (Hindle and Rooth, 1991; Sekine et al., 1992; Resnik, 1993a; Grishman and Sterling, 1994; Alshawi and Carter, 1994), or at most two case slots are dependent (Brill and Resnik, 1994; Ratnaparkhi, Reynar, and Roukos, 1994; Collins and Brooks, 1995).

2.5 Structural Disambiguation

2.5.1 The lexical approach

There have been many probabilistic methods proposed in the literature to address the structural disambiguation problem. Some methods tackle the basic problem of resolving ambiguities in quadruples (v, n_1, p, n_2) (e.g., (eat, ice-cream, with, spoon)) by mainly using lexical knowledge. Such methods can be classified into the following three types: the double approach, the triple approach, and the quadruple approach.

The first two approaches employ what I call a 'generation model' and the third approach employs what I call a 'decision model' (cf., Chapter 3).

---

while free energy functions with low temperatures *precisely* approximate it. A deterministic-annealing-based algorithm manages to find the global minimum value of the target function by continuously finding the minimum values of the free energy functions while incrementally decreasing the temperatures. (Note that deterministic annealing is different from the classical 'simulated annealing' technique (Kirkpatrick, Gelatt, and Vecchi, 1983).) In Pereira et al's case, deterministic annealing is used to find the minimum of average distortion. They have proved that, in their problem setting, minimizing average distortion is equivalent to maximizing likelihood with respect to the given data (i.e., MLE).

The double approach

This approach takes doubles of the form (v, p) and (n_1, p), like those in Table 2.4, as training data to acquire lexical knowledge and judges the attachment sites of (p, n_2) in quadruples based on the acquired knowledge.

Table 2.4: Example input data as doubles.

eat in
eat with
ice-cream with
candy with

Hindle and Rooth (1991) propose the use of the so-called 'lexical association' measure calculated based on such doubles:

$$P(p|v),$$

where random variable v represents a verb (in general a head), and random variable p a slot (preposition). They further propose viewing the disambiguation problem as that of hypothesis testing. More specifically, they calculate the 't-score,' which is a statistic on the difference between the two estimated probabilities $\hat{P}(p|v)$ and $\hat{P}(p|n_1)$:

$$t = \frac{\hat{P}(p|v) - \hat{P}(p|n_1)}{\sqrt{\frac{\hat{\sigma}_v^2}{N_v} + \frac{\hat{\sigma}_{n_1}^2}{N_{n_1}}}},$$

where $\hat{\sigma}_v$ and $\hat{\sigma}_{n_1}$ denote the standard deviations of $\hat{P}(p|v)$ and $\hat{P}(p|n_1)$, respectively, and N_v and N_{n_1} denote the data sizes used to estimate these probabilities. If, for example, $t > 1.28$, then (p, n_2) is attached to v, $t < -1.28$, (p, n_2) is attached to n_1, and otherwise no decision is made. (See also (Hindle and Rooth, 1993).)

The triple approach

This approach takes triples (v, p, n_2) and (n_1, p, n_2), i.e., case slot data, like those in Table 2.5, as training data for acquiring lexical knowledge, and performs pp-attachment disambiguation on quadruples.

For example, Resnik (1993a) proposes the use of the selectional association measure (described in Section 2) calculated on the basis of such triples. The basic idea of his method is to compare $A(n_2|v, p)$ and $A(n_2|n_1, p)$ defined in (2.3), and make a disambiguation decision.

Sekine et al. (1992) propose the use of joint probabilities $P(v, p, n_2)$ and $P(n_1, p, n_2)$ in pp-attachment disambiguation. They devised a heuristic method for estimating the probabilities. (See also (Alshawi and Carter, 1994).)

Table 2.5: Example input data as triples.

eat in park
eat with spoon
ice-cream with chocolate
eat with chopstick
candy with chocolate

The quadruple approach

This approach receives quadruples (v, n_1, p, n_2), as well as labels that indicate which way the pp-attachment goes, such as those in Table 2.6; and it learns disambiguation rules.

Table 2.6: Example input data as quadruples and labels.

eat ice-cream in park	attv
eat ice-cream with spoon	attv
eat candy with chocolate	attn

It in fact employs the conditional probability distribution – a 'decision model'

$$P(a|v, n_1, p, n_2), \tag{2.4}$$

where random variable a takes on attv and attn as its values, and random variables (v, n_1, p, n_2) take on quadruples as their values. Since the number of parameters in the distribution is very large, accurate estimation of the distribution would be impossible.

In order to address this problem, Collins and Brooks (1995) devised a back-off method. It first calculates the conditional probability $P(a|v, n_1, p, n_2)$ by using the relative frequency

$$\frac{f(a, v, n_1, p, n_2)}{f(v, n_1, p, n_2)},$$

if the denominator is larger than 0; otherwise it successively uses lower order frequencies to heuristically calculate the probability.

Ratnaparkhi, Reynar, and Roukos (1994) propose to learn the conditional probability distribution (2.4) with Maximum Entropy Estimation. They adopt the Maximum Entropy Principle (MEP) as the learning strategy, which advocates selecting the model having the maximum entropy from among the class of models that satisfies certain constraints (see Section 2.7.4 for a discussion on the relation between MDL and MEP). The fact that a model must be one such that the expected value of a feature with

respect to it equals that with respect to the empirical distribution is usually used as a constraint. Ratnaparkhi et al's method defines, for example, a feature as follows

$$f_i = \begin{cases} 1 & (p, n_2) \text{ is attached to } n_1 \text{ in (-, ice-cream, with, chocolate)} \\ 0 & \text{otherwise.} \end{cases}$$

It then incrementally selects features, and efficiently estimates the conditional distribution by using the Maximum Entropy Estimation technique (see (Jaynes, 1978; Darroch and Ratcliff, 1972; Berger, Pietra, and Pietra, 1996)).

Another method of the quadruple approach is to employ 'transformation-based error-driven learning' (Brill, 1995), as proposed in (Brill and Resnik, 1994). This method learns and uses IF-THEN type rules, where the IF parts represent conditions like (p is 'with') and (v is 'see'), and the THEN parts represent transformations from (attach to v) to (attach to n_1), and vice-versa. The first rule is always a default decision, and all the other rules indicate transformations (changes of attachment sites) subject to various IF conditions.

2.5.2 The combined approach

Although the use of lexical knowledge can effectively resolve ambiguities, it still has limitation. It is preferable, therefore, to utilize other kind of knowledge in disambiguation, especially when a decision cannot be made solely on the basis of lexical knowledge.

The following two facts suggest that syntactic knowledge should also be used for the purposes of disambiguation. First, interpretations are obtained through syntactic parsing. Second, psycholinguistists observe that there are certain syntactic principles in human's language interpretation. For example, in English a phrase on the right tends to be attached to the nearest phrase on the left, - referred to as the 'right association principle' (Kimball, 1973). (See also (Ford, Bresnan, and Kaplan, 1982; Frazier and Fodor, 1979; Hobbs and Bear, 1990; Whittemore, Ferrara, and Brunner, 1990)).

We are thus led to the problem of how to define a probability model which combines the use of both lexical semantic knowledge and syntactic knowledge. One approach is to introduce probability models on the basis of syntactic parsing. Another approach is to introduce probability models on the basis of psycholinguistic principles (Li, 1996).

Many methods belonging to the former approach have been proposed. A classical method is to employ the PCFG (Probabilistic Context Free Grammar) model (Fujisaki et al., 1989; Jelinek, Lafferty, and Mercer, 1990; Lari and Young, 1990), in which a CFG rule having the form of

$$A \rightarrow B_1 \cdots B_m$$

is associated with a conditional probability

$$P(B_1, \cdots, B_m | A). \tag{2.5}$$

In disambiguation the likelihood of an interpretation is defined as the product of the conditional probabilities of the rules which are applied in the derivation of the interpretation.

The use of PCFG, in fact, resorts more to syntactic knowledge rather than to lexical knowledge, and its performance seems to be only moderately good (Chitrao and Grishman, 1990). There are also many methods proposed which more effectively make use of lexical knowledge.

Collins (1997) proposes disambiguation through use of a generative probability model based on a lexicalized CFG (in fact, a restricted form of HPSG (Pollard and Sag, 1987)). (See also (Collins, 1996; Schabes, 1992; Hogenhout and Matsumoto, 1996; Den, 1996; Charniak, 1997).) In Collins' model, each lexicalized CFG rule is defined in the form of

$$A \rightarrow L_n \cdots L_1 H R_1 \cdots R_m,$$

where a capitalized symbol denotes a category, with H being the head category on the right hand site. A category is defined in the form of $C(w,t)$, where C denotes the name of the category, w the head word associated with the category, and t the part-of-speech tag assigned to the head word. Furthermore, each rule is assigned a conditional probability $P(L_n, \cdots, L_1, H, R_1, \cdots, R_m|A)$ (cf., (2.5)) that is assumed to satisfy

$$P(L_n, \cdots, L_1, H, R_1, \cdots, R_m|A) = P(H|A) \cdot P(L_1, \cdots, L_n|A, H) \cdot P(R_1, \cdots, R_m|A, H).$$

In disambiguation, the likelihood of an interpretation is defined as the product of the conditional probabilities of the rules which are applied in the derivation of the interpretation. While Collins has devised several heuristic methods for estimating the probability model, further investigation into learning methods for this model still appears necessary.

Magerman (1995) proposes a new parsing approach based on probabilistic decision tree models (Quinlan and Rivest, 1989; Yamanishi, 1992a) to replace conventional context free parsing. His method uses decision tree models to construct parse trees in a bottom-up and left-to-right fashion. A decision might be made, for example, to create a new parse-tree-node, and conditions for making that decision might be, for example, the appearances of certain words and certain tags in a node currently being focussed upon and in its neighbor nodes. Magerman has also devised an efficient algorithm for finding the parse tree (interpretation) with the highest likelihood value. The advantages of this method are its effective use of contextual information and its non-use of a hand-made grammar. (See also (Magerman and Marcus, 1991; Magerman, 1994; Black et al., 1993; Ratnaparkhi, 1997; Haruno, Shirai, and Ooyama, 1998))

Su and Chang (1988) propose the use of a probabilistic score function for disambiguation in generalized LR parsing (see also (Su et al., 1989; Chang, Luo, and Su, 1992; Chiang, Lin, and Su, 1995; Wright, 1990; Kita, 1992; Briscoe and Carroll, 1993;

Inui, Sornlertlamvanich, and Tanaka, 1998)). They first introduce a conditional probability of a category obtained after a reduction operation and in the context of the reduced categories and of the categories immediately left and right of those reduced categories. The score function, then, is defined as the product of the conditional probabilities appearing in the derivation of the interpretation. The advantage of the use of this score function is its context-sensitivity, which can yield more accurate results in disambiguation.

Alshawi and Carter (1994) propose for disambiguation purposes the use of a linear combination of various preference functions based on lexical and syntactic knowledge. They have devised a method for training the weights of a linear combination. Specifically, they employ the minimization of a squared-error cost function as a learning strategy and employ a 'hill-climbing' algorithm to iteratively adjust weights on the basis of training data.

Additionally, some non-probabilistic approaches to structural disambiguation have also been proposed (e.g., (Wilks, 1975; Wermter, 1989; Nagao, 1990; Kurohashi and Nagao, 1994)).

2.6 Word Sense Disambiguation

Word sense disambiguation is an issue closely related to the structural disambiguation problem. For example, when analyzing the sentence "Time flies like an arrow," we obtain a number of ambiguous interpretations. Resolving the sense ambiguity of the word 'fly' (i.e., determining whether the word indicates 'an insect' or 'the action of moving through the air'), for example, helps resolve the structural ambiguity, and the converse is true as well.

There have been many methods proposed to address the word sense disambiguation problem. (A number of tasks in natural language processing, in fact, fall into the category of word sense disambiguation (Yarowsky, 1993). These include homograph disambiguation in speech synthesis, word selection in machine translation, and spelling correction in document processing.)

A simple approach to word sense disambiguation is to employ the conditional distribution – a 'decision model'

$$P(D|E_1, \cdots, E_n),$$

where random variable D assumes word senses as its values, and random variables $E_i (i = 1, \cdots, n)$ represent pieces of evidence for disambiguation. For example, D can be the insect sense or the action sense of the word 'fly,' E_i can be the presence or absence of the word 'time' in the context. Word sense disambiguation, then, can be realized as the process of finding the sense d whose conditional probability $P(d|e_1, \cdots, e_n)$ is the largest, where $e_1, \cdots, e_n$ are the values of the random variables $E_1, \cdots, E_n$ in the current context.

Since the conditional distribution has a large number of parameters, however, it is

difficult to estimate them. One solution to this difficulty is to estimate the conditional probabilities by using Bayes' rule and by assuming that the pieces of evidence for disambiguation are mutually independent (Yarowsky, 1992). Specifically, we select a sense d satisfying:

$$\begin{aligned}\arg\max_{d\in D} P(d|e_1,\cdots,e_n) &= \arg\max_{d\in D}\{\tfrac{P(d)\cdot P(e_1,\cdots,e_n|d)}{P(e_1,\cdots,e_n)}\},\\ &= \arg\max_{d\in D}\{P(d)\cdot P(e_1,\cdots,e_n|d)\},\\ &= \arg\max_{d\in D}\{P(d)\cdot \textstyle\prod_{i=1}^{n} P(e_i|d)\},\end{aligned}$$

Another way of estimating the conditional probability distribution is to represent it in the form of a *probabilistic* decision list[5], as is proposed in (Yarowsky, 1994). Since a decision list is a sequence of IF-THEN type rules, the use of it in disambiguation turns out to utilize only the strongest pieces of evidence. Yarowsky has also devised a heuristic method for efficient learning of a probabilistic decision list. The merits of this method are ease of implementation, efficiency in processing, and clarity.

Another approach to word sense disambiguation is the use of weighted majority learning (Littlestone, 1988; Littlestone and Warmuth, 1994). Suppose, for the sake of simplicity, that the disambiguation decision is binary, i.e., it can be represented as 1 or 0. We can first define a linear threshold function:

$$\sum_{i=1}^{n} w_i \cdot x_i$$

where feature $x_i(i = 1, \cdots, n)$ takes on 1 and 0 as its values, representing the presence and absence of a piece of evidence, respectively, and $w_i(i = 1, \cdots, n)$ denotes a non-negative real-valued weight. In disambiguation, if the function exceeds a predetermined threshold θ, we choose 1, otherwise 0. We can further employ a learning algorithm called 'winnow' that updates the weights in an on-line (or incremental) fashion.[6] This algorithm has the advantage of being able to handle a large set of features, and at the same time not ordinarily be affected by features that are irrelevant to the disambiguation decision. (See (Golding and Roth, 1996).)

For word sense disambiguation methods, see also (Black, 1988; Brown et al., 1991; Guthrie et al., 1991; Gale, Church, and Yarowsky, 1992; McRoy, 1992; Leacock, Towell, and Voorhees, 1993; Yarowsky, 1993; Bruce and Wiebe, 1994; Niwa and Nitta, 1994; Voorhees, Leacock, and Towell, 1995; Yarowsky, 1995; Golding and Schabes, 1996; Ng and Lee, 1996; Fujii et al., 1996; Schütze, 1997; Schütze, 1998).

[5] A probabilistic decision list (Yamanishi, 1992a) is a kind of conditional distribution and different from a *deterministic* decision list (Rivest, 1987), which is a kind of Boolean function.

[6] Winnow is similar to the well-known classical 'perceptron' algorithm, but the former uses a multiplicative weight update scheme while the latter uses an additive weight update scheme. Littlestone (1988) has shown that winnow performs much better than perceptron when many attributes are irrelevant.

2.7 Introduction to MDL

The Minimum Description Length principle is a strategy (criterion) for data compression and statistical estimation, proposed by Rissanen (1978; 1983; 1984; 1986; 1989; 1996; 1997). Related strategies were also proposed and studied independently in (Solomonoff, 1964; Wallace and Boulton, 1968; Schwarz, 1978). A number of important properties of MDL have been demonstrated by Barron and Cover (1991) and Yamanishi (1992a).

MDL states that, for both data compression and statistical estimation, the best probability model *with respect to given data* is that which requires the shortest code length in bits for encoding the model itself and the data observed through it.[7]

In this section, we will consider the basic concept of MDL and, in particular how to calculate description length. Interested readers are referred to (Quinlan and Rivest, 1989; Yamanishi, 1992a; Yamanishi, 1992b; Han and Kobayashi, 1994) for an introduction to MDL.

2.7.1 Basics of Information Theory

IID process

Suppose that a data sequence (or a sequence of symbols)

$$x^n = x_1 x_2 \cdots x_n$$

is *independently* generated according to a discrete probability distribution

$$P(X), \tag{2.6}$$

where random variable (information source) X takes on values from a set of symbols:

$$\{1, 2, \cdots, s\}.$$

Such a data generation process is generally referred to as 'i.i.d' (independently and identically distributed).

In order to transmit or compress the data sequence, we need to define a code for encoding the information source X, i.e., to assign to each value of X a codeword, namely a bit string. In order for the decoder to be able to decode a codeword as soon as it comes to the end of that codeword, the code must be one in which no codeword is a prefix of any other codeword. Such a code is called a 'prefix code.'

[7]In this thesis, I describe MDL as a criterion for both data compression and statistical estimation. Strictly speaking, however, it is only referred to as the 'MDL principle' when used as a criterion for statistical estimation.

Theorem 1 *The sufficient and necessary condition for a code to be a prefix code is as follows,*

$$\sum_{i=1}^{s} 2^{-l(i)} \leq 1,$$

where $l(i)$ denotes the code length of the codeword assigned to symbol i.

This is known as Kraft's inequality.

We define the expected (average) code length of a code for encoding the information source X as

$$L(X) = \sum_{i=1}^{s} P(i) \cdot l(i).$$

Moreover, we define the entropy of (the distribution of) X as[8]

$$H(X) = -\sum_{i=1}^{s} P(i) \cdot \log P(i).$$

Theorem 2 *The expected code length of a prefix code for encoding the information source X is greater than or equal to the entropy of X, namely*

$$L(X) \geq H(X).$$

We can define a prefix code in which symbol i is assigned a codeword with code length

$$l(i) = -\log P(i) \quad (i = 1, \cdots, s),$$

according to Theorem 1, since

$$\sum_{i=1}^{s} 2^{\log P(i)} = 1.$$

Such a code is on average the most efficient prefix code, according to Theorem 2. Hereafter, we refer to this type of code as a ' non-redundant code.' (In real communication, a code length must be a truncated integer: $\lceil -\log P(i) \rceil$,[9] but we use here $-\log P(i)$ for ease of mathematical manipulation. This is not harmful and on average the error due to it is negligible.) When the distribution $P(X)$ is a uniform distribution, i.e.,

$$P(i) = \frac{1}{s} \quad (i = 1, \cdots, s),$$

the code length for encoding each symbol i turns out to be

$$l(i) = -\log P(i) = -\log \frac{1}{s} = \log s \quad (i = 1, \cdots, s).$$

[8]Throughout this thesis, 'log' denotes logarithm to the base 2.

[9]$\lceil x \rceil$ denotes the least integer not less than x.

General case

We next consider a more general case. We assume that the data sequence

$$x^n = x_1 x_2 \cdots x_n$$

is generated according to a probability distribution $P(X^n)$ where random variable $X_i (i = 1, \cdots, n)$ takes on values from $\{1, 2, \cdots, s\}$. The data generation process needs neither be i.i.d. nor even stationary (for the definition of a stationary process, see, for example, (Cover and Thomas, 1991)). Again, our goal is to transmit or compress the data sequence.

We define the expected code length for encoding a sequence of n symbols as

$$L(X^n) = \sum_{x^n} P(x^n) \cdot l(x^n),$$

where $P(x^n)$ represents the probability of observing the data sequence x^n and $l(x^n)$ the code length for encoding x^n. We further define the entropy of (the distribution of) X^n as

$$H(X^n) = -\sum_{x^n} P(x^n) \log P(x^n).$$

We have the following theorem, widely known as Shannon's first theorem (cf., (Cover and Thomas, 1991)).

Theorem 3 *The expected code length of a prefix code for encoding a sequence of n symbols X^n is greater than or equal to the entropy of X^n, namely*

$$L(X^n) \geq H(X^n).$$

As in the i.i.d. case, we can define a non-redundant code in which the code length for encoding the data sequence x^n is

$$l(x^n) = -\log P(x^n). \tag{2.7}$$

The expected code length of the code for encoding a sequence of n symbols then becomes

$$L(X^n) = H(X^n). \tag{2.8}$$

Here we assume that we know in advance the distribution $P(X)$ (in general $P(X^n)$). In practice, however, we usually do not know what kind of distribution it is. We have to estimate it by using the same data sequence x^n and transmit first the estimated model and then the data sequence, which leads us to the notion of two-stage coding.

2.7.2 Two-stage code and MDL

In two-stage coding, we first introduce a class of models which includes all of the possible models which can give rise to the data. We then choose a prefix code and encode each model in the class. The decoder is informed in advance as to which class has been introduced and which code has been chosen, and thus no matter which model is transmitted, the decoder will be able to identify it. We next calculate the total code length for encoding each model and the data through the model, and select the model with the shortest total code length. In actual transmission, we transmit first the selected model and then the data through the model. The decoder then can restore the data perfectly.

Model class

We first introduce a class of models, of which each consists of a discrete model (an expression) and a parameter vector (a number of parameters). When a discrete model is specified, the number of parameters is also determined.

For example, the tree cut models within a thesaurus tree, to be defined in Chapter 4, form a model class. A discrete model in this case corresponds to a cut in the thesaurus tree. The number of free parameters equals the number of nodes in the cut minus one.

The class of 'linear regression models' is also an example model class. A discrete model is

$$a_0 + a_1 \cdot x_1 + \cdots + a_k \cdot x_k + \epsilon,$$

where $x_i (i = 1, \cdots, k)$ denotes a random variable, $a_i (i = 0, 1, \cdots, k)$ a parameter, and ϵ a random variable based on the standard normal distribution $N(0, 1)$. The number of parameters in this model equals $(k + 1)$.

A class of models can be denoted as

$$\mathcal{M} = \{P_\theta(X) : \theta \in \Theta(m), m \in M\},$$

where m stands for a discrete model, M a set of discrete models, θ a parameter vector, and $\Theta(m)$ a parameter space associated with m.

Usually we assume that the model class we introduced contains the 'true' model which has given rise to the data, but it does not matter if it does not. In such case, the best model selected from the class can be considered an approximation of the true model. The model class we introduce reflects our prior knowledge on the problem.

Total description length

We next consider how to calculate total description length.

Total description length equals the sum total of the code length for encoding a discrete model (model description length $l(m)$), the code length for encoding parameters given the discrete model (parameter description length $l(\theta|m)$), and the code length

for encoding the data given the discrete model and the parameters (data description length $l(x^n|m,\theta)$). Note that we also sometimes refer to the model description length as $l(m) + l(\theta|m)$.

Our goal is to find the minimum description length of the data (in number of bits) with respect to the model class, namely,

$$L_{min}(x^n : \mathcal{M}) = \min_{m \in M} \min_{\theta \in \Theta(m)} \left(\; l(m) + l(\theta|m) + l(x^n|m,\theta) \; \right).$$

Model description length

Let us first consider how to calculate model description length $l(m)$. The choice of a code for encoding discrete models is subjective; it depends on our prior knowledge on the model class.

If the set of discrete models M is finite and the probability distribution over it is a uniform distribution, i.e.,

$$P(m) = \frac{1}{|M|}, m \in M,$$

then we need

$$l(m) = \log |M|$$

to encode each discrete model m using a non-redundant code.

If M is a countable set, i.e., each of its members can be assigned a positive integer, then the 'Elias code,' which is usually used for encoding integers, can be employed (Rissanen, 1989). Letting i be the integer assigned to a discrete model m, we need

$$l(m) = \log c + \log i + \log \log i + \cdots$$

to encode m. Here the sum includes all the positive iterates and c denotes a constant of about 2.865.

Parameter description length and data description length

When a discrete model m is fixed, a parameter space will be uniquely determined. The model class turns out to be

$$\mathcal{M}_m = \{P_\theta(X) : \theta \in \Theta\},$$

where θ denotes a parameter vector, and Θ the parameter space. Suppose that the dimension of the parameter space is k, then θ is a vector with k real-valued components:

$$\theta = (\theta_1, \cdots, \theta_k)^T,$$

where X^T denotes a transpose of X.

We next consider a way of calculating the sum of the parameter description length and the data description length through its minimization:

$$\min_{\theta \in \Theta}(l(\theta|m) + l(x^n|m, \theta)).$$

Since the parameter space Θ is usually a subspace of the k-dimensional Euclidean space and has an infinite number of points (parameter vectors), straightforwardly encoding each point in the space takes the code length to infinity, and thus is intractable. (Recall the fact that before transmitting an element in a set, we need encode each element in the set.) One possible way to deal with this difficulty is to discretize the parameter space; the process can be defined as a mapping from the parameter space to a discretized space, depending on the data size n:

$$\Delta_n : \Theta \rightarrow \Theta_n.$$

A discretized parameter space consists of a finite number of elements (cells). We can designate one point in each cell as its representative and use only the representatives for encoding parameters. The minimization then turns out to be

$$\min_{\bar{\theta} \in \Theta_n}(l(\bar{\theta}|m) + l(x^n|m, \bar{\theta})),$$

where $l(\bar{\theta}|m)$ denotes the code length for encoding a representative $\bar{\theta}$ and $l(x^n|m, \bar{\theta})$ denotes the code length for encoding the data x^n through that representative.

A simple way of conducting discretization is to define a cell as a micro k-dimensional rectangular solid having length δ_i on the axis of θ_i. If the volume of the parameter space is V, then we have $V/(\delta_1 \cdots \delta_k)$ number of cells. If the distribution over the cells is uniform, then we need

$$l(\bar{\theta}|m) = \log \frac{V}{\delta_1 \cdots \delta_k}$$

to encode each representative $\bar{\theta}$ using a non-redundant code.

On the other hand, since the number of parameters is fixed and the data is given, we can estimate the parameters by employing Maximum Likelihood Estimation (MLE), obtaining

$$\hat{\theta} = (\hat{\theta}_1, \cdots, \hat{\theta}_k)^T.$$

We may expect that the representative of the cell into which the maximum likelihood estimate falls is the nearest to the true parameter vector among all representatives. And thus, instead of conducting minimization over all representatives, we need only consider minimization with respect to the representative of the cell which the maximum likelihood estimate belongs to. This representative is denoted here as $\tilde{\theta}$. We approximate the difference between $\hat{\theta}$ and $\tilde{\theta}$ as

$$\tilde{\theta} - \hat{\theta} \approx \delta \qquad \delta = (\delta_1, \cdots, \delta_k)^T.$$

Data description length using $\tilde{\theta}$ then becomes

$$l(x^n|m, \tilde{\theta}) = -\log P_{\tilde{\theta}}(x^n).$$

Now, we need consider only

$$\min_\delta(l(x^n|m, \tilde{\theta}) + l(\tilde{\theta}|m)) = \min_\delta \left(\log \tfrac{V}{\delta_1 \cdots \delta_k} - \log P_{\tilde{\theta}}(x^n)\right).$$

There is a trade-off relationship between the first term and the second term. If δ is large, then the first term will be small, while on the other hand the second term will be large, and vice-versa. That means that if we discretize the parameter space loosely, we will need less code length for encoding the parameters, but more code length for encoding the data. On the other hand, if we discretize the parameter space precisely, then we need less code length for encoding the data, but more code length for encoding the parameters.

In this way of calculation (see Appendix A.1 for a derivation), we have

$$l(\theta|m) + l(x^n|m, \theta) = -\log P_{\hat{\theta}}(x^n) + \frac{k}{2} \cdot \log n + O(1), \tag{2.9}$$

where $O(1)$ indicates $\lim_{n\to\infty} O(1) = c$, a constant. The first term corresponds to the data description length and has the same form as that in (2.7). The second term corresponds to the parameter description length. An intuitive explanation of it is that the standard deviation of the maximum likelihood estimator of one of the parameters is of order $O(\frac{1}{\sqrt{n}})$,[10] and hence encoding the parameters using more than $k \cdot (-\log \frac{1}{\sqrt{n}})$ $= \frac{k}{2} \cdot \log n$ bits would be wasteful for the given data size.

In this way, the sum of the two kinds of description length $l(\theta|m) + l(x^n|m, \theta)$ is obtained for a fixed discrete model m (and a fixed dimension k). For a different m, the sum can also be calculated.

Selecting a model with minimum total description length

Finally, the minimum total description length becomes, for example,

$$L_{min}(x^n : \mathcal{M}) = \min_m \left(-\log P_{\hat{\theta}}(x^n) + \frac{k}{2} \cdot \log n + \log |M|\right).$$

We select the model with the minimum total description length for transmission (data compression).

[10] It is well known that under certain suitable conditions, when the data size increases, the distribution of the maximum likelihood estimator $\hat{\theta}$ will asymptotically become the normal distribution $N(\theta^*, \frac{1}{n \cdot I})$ where θ^* denotes the true parameter vector, I the Fisher information matrix, and n the data size (Fisher, 1956).

2.7.3 MDL as data compression criterion

Rissanen has proved that MDL is an optimal criterion for data compression.

Theorem 4 (Rissanen, 1984) *Under certain suitable conditions, the expected code length of the two-stage code described above (with code length (2.9) for encoding the data sequence x^n) satisfies*

$$L(X^n) = H(X^n) + \frac{k}{2} \cdot \log n + O(1),$$

where $H(X^n)$ denotes the entropy of X^n.

This theorem indicates that when we do not know the true distribution $P(X^n)$ in communication, we have to waste on average about $\frac{k}{2} \log n$ bits of code length (cf., (2.8)).

Theorem 5 (Rissanen, 1984) *Under certain suitable conditions, for* any prefix code*, for some $\epsilon_n > 0$ such that* $\lim_{n\to\infty} \epsilon_n = 0$ *and $\Omega_n \subset \Theta$ such that the volume of it* $\mathrm{vol}(\Omega_n)$ *satisfies* $\lim_{n\to\infty} \mathrm{vol}(\Omega_n) = 0$*, for any model with parameter $\theta \in (\Theta - \Omega_n)$, the expected code length $L(X^n)$ is bounded from below by*

$$L(X^n) \geq H(X^n) + \left(\frac{k}{2} - \epsilon_n\right) \cdot \log n.$$

This theorem indicates that in general, i.e., excluding some special cases, we cannot make the average code length of a prefix code more efficient than the quantity $H(X^n) + \frac{k}{2} \cdot \log n$. The introduction of Ω_n eliminates the case in which we happen to select the true model and achieve on average a very short code length: $H(X^n)$. Theorem 5 can be considered as an extension of Shannon's Theorem (Theorem 3).

Theorems 4 and 5 suggest that using the two-stage code above is nearly optimal in terms of *expected* code length. We can, therefore, say that encoding a data sequence x^n in the way described in (2.9) is the most efficient approach not only to encoding the data sequence, but also, on average, to encoding a sequence of n symbols.

2.7.4 MDL as estimation criterion

The MDL principle stipulates that selecting a model having the minimum description length is also optimal for conducting statistical estimation that includes model selection.

Definition of MDL

The MDL principle can be described more formally as follows (Rissanen, 1989; Barron and Cover, 1991). For a data sequence x^n and for a model class $\mathcal{M} = \{P_\theta(X) : \theta \in$

$\Theta(m), m \in M\}$, the minimum description length of the data with respect to the class is defined as

$$L_{min}(x^n : \mathcal{M}) = \min_{m \in M} \inf_{\Delta_n} \min_{\bar{\theta} \in \Theta_n} \left\{ -\log P_{\bar{\theta}}(x^n) + l(\bar{\theta}|m) + l(m) \right\}, \qquad (2.10)$$

where $\Delta_n : \Theta(m) \to \Theta_n$ denotes a discretization of $\Theta(m)$ and where $l(\bar{\theta}|m)$ is the code length for encoding $\bar{\theta} \in \Theta_n$, satisfying

$$\sum_{\bar{\theta} \in \Theta_n} 2^{-l(\bar{\theta}|m)} \leq 1.$$

Note that '$\inf_{\Delta_n}$' stead of '$\min_{\Delta_n}$' is used here because there are an infinite number of points which can serve as a representative for a cell. Furthermore, $l(m)$ is the code length for encoding $m \in M$, satisfying

$$\sum_{m \in M} 2^{-l(m)} \leq 1.$$

For both data compression and statistical estimation, the best probability model *with respect to the given data* is that which achieves the minimum description length given in (2.10).

The minimum description length defined in (2.10) is also referred to as the 'stochastic complexity' of the data relative to the model class.

Advantages

MDL offers many advantages as a criterion for statistical estimation, the most important perhaps being its optimal convergency rate.

Consistency

The models estimated by MDL converge *with probability one* to the true model when data size increases – a property referred to as 'consistency' (Barron and Cover, 1991). That means that not only the parameters themselves but also the number of them converge to those of the true model.

Rate of convergence

Consistency, however, is a characteristic to be considered only when data size is large; in practice, when data size can generally be expected to be small, rate of convergence is a more important guide to the performance of an estimator.

Barron and Cover (1991) have verified that MDL as an estimation strategy is near optimal in terms of the rate of convergence of its estimated models to the *true* model as the data size increases. *When the true model is included in the class of models considered*, the models selected by MDL converge in probability to the true model at

the rate of $O(\frac{k^* \cdot \log n}{2 \cdot n})$, where k^* is the number of parameters in the true model, and n the data size. This is nearly optimal.

Yamanishi (1992a) has derived an upper bound on the data size necessary for learning *probably approximately correctly* (PAC) a model from among a class of conditional distributions, which he calls stochastic rules with finite partitioning. This upper bound is of order $O(\frac{k^*}{\epsilon} \log \frac{k^*}{\epsilon} + \frac{l(m)}{\epsilon})$, where k^* denotes the number of parameters of the true model, and $\epsilon (0 < \epsilon < 1)$ the accuracy parameter for the stochastic PAC learning. For MLE, the corresponding upper bound is of order $O(\frac{k_{max}}{\epsilon} \log \frac{k_{max}}{\epsilon} + \frac{l(m)}{\epsilon})$, where k_{max} denotes the maximum of the number of parameters of a model in the model class. These upper bounds indicate that MDL requires less data than MLE to achieve the same accuracy in statistical learning, provided that $k_{max} > k^*$ (note that, in general, $k_{max} \geq k^*$).

MDL and MLE

When the number of parameters in a probability model is fixed, and the estimation problem involves only the estimation of parameters, MLE is known to be satisfactory (Fisher, 1956). Furthermore, for such a fixed model, it is known that MLE is equivalent to MDL: given the data $x^n = x_1 \cdots x_n$, the maximum likelihood estimator $\hat{\theta}$ is defined as one that maximizes likelihood with respect to the data, that is,

$$\hat{\theta} = \arg\max_{\theta} P(x^n). \tag{2.11}$$

It is easy to see that $\hat{\theta}$ also satisfies

$$\hat{\theta} = \arg\min_{\theta} - \log P(x^n).$$

This is, in fact, no more than the MDL estimator in this case, since $-\log P_{\hat{\theta}}(x^n)$ is the *data description length*.

When the estimation problem involves model selection, MDL's behavior significantly deviates from that of MLE. This is because MDL insists on minimizing the sum total of the data description length *and* the model description length, while MLE is still equivalent to minimizing the data description length alone. We can, therefore, say that MLE is a special case of MDL.

Note that in (2.9), the first term is of order $O(n)$ and the second term is of order $O(\log n)$, and thus the first term will dominate that formula when the data size increases. That means that when data size is sufficiently large, the MDL estimate will turn out to be the MLE estimate; otherwise the MDL estimate will be different from the MLE estimate.

MDL and Bayesian Estimation

In an interpretation of MDL from the viewpoint of Bayesian Estimation, MDL is essentially equivalent to the 'MAP estimation' in Bayesian terminology. Given data

D and a number of models, the Bayesian (MAP) estimator $\hat{M}$ is defined as one that maximizes posterior probability, i.e.,

$$\begin{aligned} \hat{M} &= \arg\max_M (P(M|D)) \\ &= \arg\max_M (\tfrac{P(M)\cdot P(D|M)}{P(D)}) \\ &= \arg\max_M (P(M)\cdot P(D|M)), \end{aligned} \tag{2.12}$$

where $P(M)$ denotes the prior probability of model M and $P(D|M)$ the probability of observing data D through M. In the same way, $\hat{M}$ satisfies

$$\hat{M} = \arg\min_M (-\log P(M) - \log P(D|M)).$$

This is equivalent to the MDL estimator if we take $-\log P(M)$ to be the model description length. Interpreting $-\log P(M)$ as the model description length translates, in Bayesian Estimation, to assigning larger prior probabilities to simpler models, since it is equivalent to assuming that $P(M) = (\frac{1}{2})^{l(M)}$, where $l(M)$ is the code length of model M. (Note that if we assign uniform prior probability to all models, then (2.12) becomes equivalent to (2.11), giving the MLE estimator.)

MDL and MEP

The use of the Maximum Entropy Principle (MEP) has been proposed in statistical language processing (Ratnaparkhi, Reynar, and Roukos, 1994; Ratnaparkhi, Reynar, and Roukos, 1994; Ratnaparkhi, 1997; Berger, Pietra, and Pietra, 1996; Rosenfeld, 1996)). Like MDL, MEP is also a learning criterion, one which stipulates that from among the class of models that satisfies certain constraints, the model which has the maximum entropy should be selected. Selecting a model with maximum entropy is, in fact, equivalent to selecting a model with minimum description length (Rissanen, 1983). Thus, MDL provides an information-theoretic justification of MEP.

MDL and stochastic complexity

The sum of parameter description length and data description length given in (2.9) is still a loose approximation. Recently, Rissanen has derived this more precise formula:

$$l(\theta|m) + l(x^n|m,\theta) = -\log P_{\hat{\theta}}(x^n) + \frac{k}{2}\cdot\log\frac{n}{2\pi} + \log\int\sqrt{|I(\theta)|}d\theta + o(1), \tag{2.13}$$

where $I(\theta)$ denotes the Fisher information matrix, $|\mathbf{A}|$ the determinant of matrix $\mathbf{A}$, and π the circular constant, and $o(1)$ indicates $\lim_{n\to\infty} o(1) = 0$. It is thus preferable to use this formula in practice.

This formula can be obtained not only on the basis of the 'complete two-stage code,' but also on that of 'quantized maximum likelihood code,' and has been proposed as the new definition of stochastic complexity (Rissanen, 1996). (See also (Clarke and Barron, 1990).)

When the data generation process is i.i.d. and the distribution is a discrete probability distribution like that in (2.6), the sum of parameter description length and data description length turns out to be (Rissanen, 1997)

$$l(\theta) + l(x^n|m,\theta) \quad = -\sum_{i=1}^{n} \log P_{\hat{\theta}}(x_i) + \frac{k}{2} \cdot \log \frac{n}{2 \cdot \pi} + \log \frac{\pi^{(k+1)/2}}{\Gamma(\frac{(k+1)}{2})} + o(1), \qquad (2.14)$$

where Γ denotes the Gamma function[11]. This is because in this case, the determinant of the Fisher information matrix becomes $\frac{1}{\prod_{i=1}^{k+1} P(i)}$, and the integral of its square root can be calculated by the Dirichlet's integral as $\frac{\pi^{(k+1)/2}}{\Gamma(\frac{(k+1)}{2})}$.

2.7.5 Employing MDL in NLP

Recently MDL and related techniques have become popular in natural language processing and related fields; a number of learning methods based on MDL have been proposed for various applications (Ellison, 1991; Ellison, 1992; Cartwright and Brent, 1994; Stolcke and Omohundro, 1994; Brent, Murthy, and Lundberg, 1995; Ristad and Thomas, 1995; Brent and Cartwright, 1996; Grunwald, 1996).

Coping with the data sparseness problem

MDL is a powerful tool for coping with the data sparseness problem, an inherent difficulty in statistical language processing. In general, a complicated model might be suitable for representing a problem, but it might be difficult to learn due to the sparseness of training data. On the other hand, a simple model might be easy to learn, but it might be not rich enough for representing the problem. One possible way to cope with this difficulty is to introduce a class of models with various complexities and to employ MDL to select the model having the most appropriate level of complexity.

An especially desirable property of MDL is that it takes data size into consideration. Classical statistics actually assume implicitly that the data for estimation are always sufficient. This, however, is patently untrue in natural language. Thus, the use of MDL might yield more reliable results in many NLP applications.

Employing efficient algorithms

In practice, the process of finding the optimal model in terms of MDL is very likely to be intractable because a model class usually contains too many models to calculate a description length for each of them. Thus, when we have modelized a natural language acquisition problem on the basis of a class of probability models and want to employ MDL to select the best model, what is necessary to consider next is how to perform the task efficiently, in other words, how to develop an efficient algorithm.

[11] Euler's Gamma function is defined as $\Gamma(x) = \int_0^\infty t^{x-1} \cdot e^{-t} dt$.

When the model class under consideration is restricted to one related to a tree structure, for instance, the dynamic programming technique is often applicable and the optimal model can be efficiently found. Rissanen (1997), for example, has devised such an algorithm for learning a decision tree.

Another approach is to calculate approximately the description lengths for the probability models, by using a computational-statistic technique, e.g., the Markov chain Monte-Carlo method, as is proposed in (Yamanishi, 1996).

In this thesis, I take the approach of restricting a model class to a simpler one (i.e., reducing the number of models to consider) when doing so is still reasonable for tackling the problem at hand.

Chapter 3

Models for Lexical Knowledge Acquisition

The world as we know it is our interpretation of the observable facts in the light of theories that we ourselves invent.

- Immanuel Kant (paraphrase)

In this chapter, I define probability models for each subproblem of the lexical semantic knowledge acquisition problem: (1) the hard case slot model and the soft case slot model; (2) the word-based case frame model, the class-based case frame model, and the slot-based case frame model; and (3) the hard co-occurrence model and the soft co-occurrence model. These are respectively the probability models for (1) case slot generalization, (2) case dependency learning, and (3) word clustering.

3.1 Case Slot Model

Hard case slot model

We can assume that case slot data for a case slot for a verb like that shown in Table 2.2 are generated according to a conditional probability distribution, which specifies the conditional probability of a noun given the verb and the case slot. I call such a distribution a 'case slot model.'

When the conditional probability of a noun is defined as that of the noun class to which the noun belongs, divided by the size of the noun class, I call the case slot model a 'hard-clustering-based case slot model,' or simply a 'hard case slot model.'

Suppose that $\mathcal{N}$ is the set of nouns, $\mathcal{V}$ is the set of verbs, and $\mathcal{R}$ is the set of slot names. A partition Π of $\mathcal{N}$ is defined as a set satisfying $\Pi \subseteq 2^{\mathcal{N}}$,[1] $\cup_{C \in \Pi} C = \mathcal{N}$ and

[1] 2^A denotes the power set of a set A; if, for example, $A = \{a, b\}$, then $2^A = \{\{\}, \{a\}, \{b\}, \{a, b\}\}$.

$\forall C_i, C_j \in \Pi, C_i \cap C_j = \emptyset, (i \neq j)$. An element C in Π is referred to as a 'class.' A hard case slot model with respect to a partition Π is defined as a conditional probability distribution:

$$P(n|v,r) = \frac{1}{|C|} \cdot P(C|v,r) \quad n \in C, \tag{3.1}$$

where random variable n assumes a value from $\mathcal{N}$, random variable v from $\mathcal{V}$, and random variable r from $\mathcal{R}$, and where $C \in \Gamma$ is satisfied. [2]

We can formalize the case slot generalization problem as that of estimating a hard case slot model. The problem, then, turns out to be that of selecting a model, from a class of hard case slot models, which is most likely to have given rise to the case slot data.

This formalization of case slot generalization will make it possible to deal with the data sparseness problem, an inherent difficulty in a statistical approach to natural language processing. Since many words in natural language are synonymous, it is natural to classify them into the same word class and employ class-based probability models. A class-based model usually has far fewer parameters than a word-based model, and thus the use of it can help handle the data sparseness problem. An important characteristic of the approach taken here is that it automatically conducts the optimization of word clustering by means of statistical model selection. That is to say, neither the number of word classes nor the way of word classification are determined in advance, but are determined automatically on the basis of the input data.

The uniform distribution assumption in the hard case slot model seems to be necessary for dealing with the data sparseness problem. If we were to assume that the distribution of words (nouns) within a class is a word-based distribution, then the number of parameters would not be reduced and the data sparseness problem would still prevail.

Under the uniform distribution assumption, generalization turns out to be the process of finding the best configuration of classes such that the words in each class are equally likely to be the value of the slot in question. (Words belonging to a single word class should be similar in terms of likelihood; they do not necessarily have to be synonyms.) Conversely, if we take the generalization to be such a process, then viewing it as statistical estimation of a hard case slot model seems to be quite appropriate, because the class of hard case slot models contains all of the possible models for the purposes of generalization. The word-based case slot model (i.e., one in which each word forms its own word class) is a (discrete) hard case slot model, and any grouping of words (nouns) leads to one (discrete) hard case slot model.

---

[2] Rigorously, a hard case slot model with respect to a noun partition Π should be represented as

$$P_\Pi(n|v,r) = \sum_{C \in \Pi} P(n|C) \cdot P(C|v,r)$$
$$P(n|C) = \begin{cases} \frac{1}{|C|} & n \in C \\ 0 & \text{otherwise.} \end{cases}$$

Soft case slot model

Note that in the hard case slot model a word (noun) is assumed to belong to a single class. In practice, however, many words have sense ambiguities and a word can belong to several different classes, e.g., 'bird' is a member of both $\langle$animal$\rangle$ and $\langle$meat$\rangle$. It is also possible to extend the hard case slot model so that each word probabilistically belongs to several different classes, which would allow us to resolve both syntactic and word sense ambiguities at the same time. Such a model can be defined in the form of a 'finite mixture model,' which is a linear combination of the word probability distributions within individual word (noun) classes. I call such a model a 'soft-clustering-based case slot model,' or simply a 'soft case slot model.'

First, a covering Γ of the noun set $\mathcal{N}$ is defined as a set satisfying $\Gamma \subseteq 2^{\mathcal{N}}$, $\cup_{C \in \Gamma} C = \mathcal{N}$. An element C in Γ is referred to as a 'class.' A soft case slot model with respect to a covering Γ is defined as a conditional probability distribution:

$$P(n|v,r) = \sum_{C \in \Gamma} P(C|v,r) \cdot P(n|C) \tag{3.2}$$

where random variable n denotes a noun, random variable v a verb, and random variable r a slot name. We can also formalize the case slot generalization problem as that of estimating a soft case slot model.

If we assume, in a soft case slot model, that a word can only belong to a single class alone and that the distribution within a class is a uniform distribution, then the soft case slot model will become a hard case slot model.

Numbers of parameters

Table 3.1 shows the numbers of parameters in a word-based case slot model (2.1), a hard case slot model (3.1), and a soft case slot model (3.2). Here N denotes the size of the set of nouns, Π the partition in the hard case slot model, and Γ the covering in the soft case slot model.

Table 3.1: Numbers of parameters in case slot models.

word-based model	$O(N)$
hard case slot model	$O(\|\Pi\|)$
soft case slot model	$O(\|\Gamma\| + \sum_{C \in \Gamma} \|C\|)$

The number of parameters in a hard case slot models is generally smaller than that in a soft case slot model. Furthermore, the number of parameters in a soft case slot model is generally smaller than that in a word-based case slot model (note that the parameters $P(n|C)$ is common to each soft case slot model). As a result, hard case slot models require less data for parameter estimation than soft case slot models, and

soft case slot models less data than word-based case slot models. That is to say, hard and soft case slot models are more useful than word-based models, given the fact that usually the size of data for training is small.

Unfortunately, currently available data sizes are still insufficient for the accurate estimating of a soft case slot model. (Appendix A.2 shows a method for learning a soft case slot model.) (See (Li and Yamanishi, 1997) for a method of using a finite mixture model in document classification, for which more data are generally available.)

In this thesis, I address only the issue of estimating a hard case slot model. With regard to the word-sense ambiguity problem, one can employ an existing word-sense disambiguation technique (cf., Chapter2) in pre-processing, and use the disambiguated word senses as virtual words in the subsequent learning process.

3.2 Case Frame Model

We can assume that case frame data like that in Table 2.1 are generated according to a multi-dimensional discrete joint probability distribution in which random variables represent case slots. I call such a distribution a 'case frame model.' We can formalize the case dependency learning problem as that of estimating a case frame model. The dependencies between case slots are represented as *probabilistic* dependencies between random variables. (Recall that random variables $X_1, \cdots, X_n$ are *mutually* independent, if for any $k \leq n$, and any $1 \leq i_1 < \cdots < i_k \leq n$, $P(X_{i_1}, \cdots, X_{i_k}) = P(X_{i_1}) \cdots P(X_{i_k})$; otherwise, they are *mutually* dependent.)

The case frame model is the joint probability distribution of type,

$$P_Y(X_1, X_2, \cdots, X_n),$$

where index Y stands for the verb, and each of the random variables $X_i, i = 1, 2, \cdots, n$, represents a case slot.

In this thesis, 'case slots' refers to *surface* case slots, but they can also be *deep* case slots. Furthermore, obligatory cases and optional cases are uniformly treated. The possible case slots can vary from verb to verb. They can also be a predetermined set for all of the verbs, with most of the slots corresponding to (English) prepositions.

The case frame model can be further classified into three types of probability models according to the type of value each random variable X_i assumes. When X_i assumes a word or a special symbol 0 as its value, the corresponding model is referred to as a 'word-based case frame model.' Here 0 indicates the absence of the case slot in question. When X_i assumes a word-class (such as $\langle$person$\rangle$ or $\langle$company$\rangle$) or 0 as its value, the corresponding model is referred to as a 'class-based case frame model.' When X_i takes on 1 or 0 as its value, the model is called a 'slot-based case frame model.' Here 1 indicates the presence of the case slot in question, and 0 the absence of it. For example, the data in Table 3.2 could have been generated by a word-based model, the data in Table 3.3 by a class-based model, where $\langle \cdots \rangle$ denotes a word class, and the

Table 3.2: Example case frame data generated by a word-based model.

Case frame	Frequency
(fly (arg1 girl)(arg2 jet))	2
(fly (arg1 boy)(arg2 helicopter))	1
(fly (arg1 company)(arg2 jet))	2
(fly (arg1 girl)(arg2 company))	1
(fly (arg1 boy)(to Tokyo))	1
(fly (arg1 girl)(from Tokyo) (to New York))	1
(fly (arg1 JAL)(from Tokyo) (to Bejing))	1

Table 3.3: Example case frame data generated by a class-based model.

Case frame	Frequency
(fly (arg1 ⟨person⟩)(arg2 ⟨airplane⟩))	3
(fly (arg1 ⟨company⟩)(arg2 ⟨airplane⟩))	2
(fly (arg1 ⟨person⟩)(arg2 ⟨company⟩))	1
(fly (arg1 ⟨person⟩)(to ⟨place⟩))	1
(fly (arg1 ⟨person⟩)(from ⟨place⟩)(to ⟨place⟩))	1
(fly (arg1 ⟨company⟩)(from ⟨place⟩)(to ⟨place⟩))	1

data in Table 3.4 by a slot-based model. Suppose, for simplicity, that there are only 4 possible case slots corresponding, respectively, to subject, direct object, 'from' phrase, and 'to' phrase. Then,

$$P_{\text{fly}}(X_{\text{arg1}} = \text{girl}, X_{\text{arg2}} = \text{jet}, X_{\text{from}} = 0, X_{\text{to}} = 0)$$

is specified by a word-based case frame model. In contrast,

$$P_{\text{fly}}(X_{\text{arg1}} = \langle\text{person}\rangle, X_{\text{arg2}} = \langle\text{airplane}\rangle, X_{\text{from}} = 0, X_{\text{to}} = 0)$$

is specified by a class-based case frame model, where ⟨person⟩ and ⟨airplane⟩ denote

Table 3.4: Example case frame data generated by a slot-based model.

Case frame	Frequency
(fly (arg1 1)(arg2 1))	6
(fly (arg1 1)(to 1))	1
(fly (arg1 1)(from 1)(to 1))	2

word classes. Finally,

$$P_{\text{fly}}(X_{\text{arg1}} = 1, X_{\text{arg2}} = 1, X_{\text{from}} = 0, X_{\text{to}} = 0)$$

is specified by a slot-based case frame model. One can also define a combined model in which, for example, some random variables assume word classes and 0 as their values while others assume 1 and 0.

Note that since in general

$$\begin{aligned}&P_{\text{fly}}(X_{\text{arg1}} = 1, X_{\text{arg2}} = 1, X_{\text{from}} = 0, X_{\text{to}} = 0)\\ &\neq P_{\text{fly}}(X_{\text{arg1}} = 1, X_{\text{arg2}} = 1),\end{aligned}$$

one should not use here the joint probability $P_{\text{fly}}(X_{\text{arg1}} = 1, X_{\text{arg2}} = 1)$ as the probability of the case frame '(fly (arg1 1)(arg2 1)).'

In learning and using of the case frame models, it is also assumed that word sense ambiguities have been resolved in pre-processing.

One may argue that when the ambiguities of a verb are resolved, there would not exist case dependencies at all (cf., 'fly' in sentences of (1.2)). Sense ambiguities, however, are generally difficult to define precisely. I think that it is preferable not to resolve them until doing so is necessary in a particular application. That is to say, I think that, in general, case dependencies do exist and the development of a method for learning them is needed.

Numbers of parameters

Table 3.5 shows the numbers of parameters in a word-based case frame model, a class-based case frame model, and a slot-based case frame model, where n denotes the number of random variables, N the size of the set of nouns, and k_{max} the maximum number of classes in any slot.

Table 3.5: Numbers of parameters in case frame models.

word-based case frame model	$O(N^n)$
class-based case frame model	$O(k_{max}^n)$
slot-based case frame model	$O(2^n)$

3.3 Co-occurrence Model

Hard co-occurrence model

We can assume that co-occurrence data over a set of nouns and a set of verbs like that in Figure 2.3 are generated according to a joint probability distribution that specifies

the co-occurrence probabilities of noun verb pairs. I call such a distribution a 'co-occurrence model.'

I call the co-occurrence model a 'hard-clustering-based co-occurrence model,' or simply a 'hard co-occurrence model,' when the joint probability of a noun verb pair can be defined as the product of the joint probability of the noun class and the verb class to which the noun and the verb respectively belong, the conditional probability of the noun given its noun class, and the conditional probability of the verb given its verb class.

Suppose that $\mathcal{N}$ is the set of nouns, and $\mathcal{V}$ is the set of verbs. A partition Π_n of $\mathcal{N}$ is defined as a set which satisfies $\Pi_n \subseteq 2^{\mathcal{N}}$, $\cup_{C_n \in \Pi_n} C_n = \mathcal{N}$ and $\forall C_i, C_j \in \Pi_n, C_i \cap C_j = \emptyset, (i \neq j)$. A partition Π_v of $\mathcal{V}$ is defined as a set which satisfies $\Pi_v \subseteq 2^{\mathcal{V}}$, $\cup_{C_v \in \Pi_v} C_v = \mathcal{V}$ and $\forall C_i, C_j \in \Pi_v, C_i \cap C_j = \emptyset, (i \neq j)$. Each element in a partition forms a 'class' of words. I define a hard co-occurrence model with respect to a noun partition Π_n and a verb partition Π_v as a joint probability distribution of type:

$$P(n, v) = P(C_n, C_v) \cdot P(n|C_n) \cdot P(v|C_v) \quad n \in C_n, v \in C_v, \tag{3.3}$$

where random variable n denotes a noun and random variable v a verb and where $C_n \in \Pi_n$ and $C_v \in \Pi_v$ are satisfied. [3] Figure 3.1 shows a hard co-occurrence model, one that can give rise to the co-occurrence data in Figure 2.3.

Estimating a hard co-occurrence model means selecting, from the class of such models, one that is most likely to have given rise to the co-occurrence data. The selected model will contain a hard clustering of words. We can therefore formalize the problem of word clustering as that of estimating a hard co-occurrence model.

We can restrict the hard co-occurrence model by assuming that words within a same class are generated with an equal probability (Li and Abe, 1996; Li and Abe, 1997), obtaining

$$P(n, v) = P(C_n, C_v) \cdot \frac{1}{|C_n|} \cdot \frac{1}{|C_v|} \quad n \in C_n, v \in C_v.$$

Employing such a restricted model in word clustering, however, has an undesirable tendency to result in classifying into different classes those words that have similar co-occurrence patterns but have different absolute frequencies.

[3] Rigorously, a hard co-occurrence model with respect to a noun partition Π_n and a verb partition Π_v should be represented as

$$\begin{array}{l} P_{\Pi_n \Pi_v}(n, v) = \sum_{C_n \in \Pi_n, C_v \in \Pi_v} P(C_n, C_v) \cdot P(n|C_n) \cdot P(v|C_v) \\ P(x|C_x) = \begin{cases} Q(x|C_x) & x \in C_x \\ 0 & \text{otherwise} \end{cases} \quad (x = n, v) \\ \forall C_x, \sum_{x \in C_x} Q(x|C_x) = 1, (x = n, v). \end{array}$$

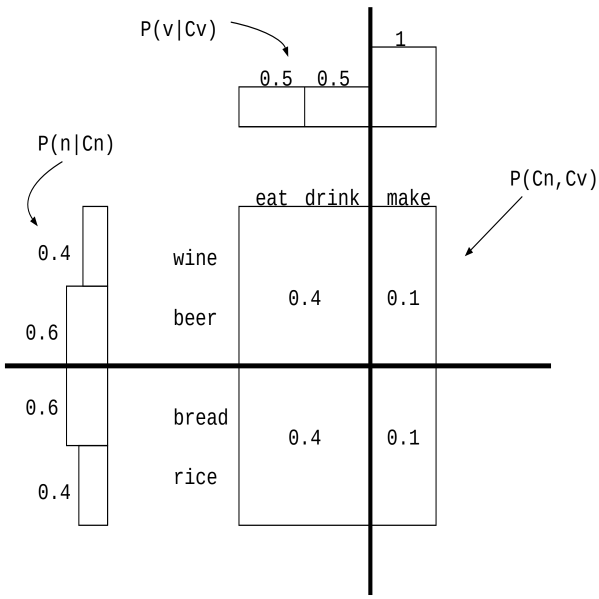


Figure 3.1: An example hard co-occurrence model.

The hard co-occurrence model in (3.3) can also be considered an extension of that proposed in (Brown et al., 1992). First, dividing the equation by $P(v)$, we obtain

$$\frac{P(n,v)}{P(v)} = P(C_n|C_v) \cdot P(n|C_n) \cdot \left(\frac{P(C_v) \cdot P(v|C_v)}{P(v)}\right) \qquad n \in C_n, v \in C_v.$$

Since $\frac{P(C_v) \cdot P(v|C_v)}{P(v)} = 1$ holds, we have

$$P(n|v) = P(C_n|C_v) \cdot P(n|C_n) \qquad n \in C_n, v \in C_v.$$

We can rewrite the model for word sequence predication as

$$P(w_2|w_1) = P(C_2|C_1) \cdot P(w_2|C_2) \qquad w_1 \in C_1, w_2 \in C_2, \tag{3.4}$$

where random variables w_1 and w_2 take on words as their values. In this way, the hard co-occurrence model turns out to be a bigram class model and is similar to that proposed in (Brown et al., 1992) (cf., Chapter 2).[4] The difference is that the model in (3.4) assumes that the configuration of word groups for C_2 and the configuration of word groups for C_1 can be different, while Brown et al's model assumes that the configurations for the two are always the same.

[4]Strictly speaking, the bigram class model proposed by (Brown et al., 1992) and the hard case slot model defined here are different types of probability models; the former is a conditional distribution, while the latter is a joint distribution.

Soft co-occurrence model

The co-occurrence model can also be defined as a double mixture model, which is a double linear combination of the word probability distributions within individual noun classes and those within individual verb classes. I call such a model a 'soft-clustering-based co-occurrence model,' or simply 'soft co-occurrence model.'

First, a covering Γ_n of the noun set $\mathcal{N}$ is defined as a set which satisfies $\Gamma_n \subseteq 2^{\mathcal{N}}$, $\cup_{C_n \in \Gamma_n} C_n = \mathcal{N}$. A covering Γ_v of the verb set $\mathcal{V}$ is defined as a set which satisfies $\Gamma_v \subseteq 2^{\mathcal{V}}$, $\cup_{C_v \in \Gamma_v} C_v = \mathcal{V}$. Each element in a covering is referred to as a 'class.' I define a soft co-occurrence model with respect to a noun covering Γ_n and a verb covering Γ_v as a joint probability distribution of type:

$$P(n, v) = \sum_{C_n \in \Gamma_n} \sum_{C_v \in \Gamma_v} P(C_n, C_v) \cdot P(n|C_n) \cdot P(v|C_v),$$

where random variable n denotes a noun and random variable v a verb. Obviously, the soft co-occurrence model includes the hard co-occurrence model as a special case.

If we assume that a verb class consists of a single verb alone, i.e., $\Gamma_v = \{\{v\}|v \in \mathcal{V}\}$, then the soft co-occurrence model turns out to be

$$P(n, v) = \sum_{C_n \in \Gamma_n} P(C_n, v) \cdot P(n|C_n),$$

which is equivalent to that proposed in (Pereira, Tishby, and Lee, 1993).

Estimating a soft co-occurrence model, thus, means selecting, from the class of such models, one that is most likely to have given rise to the co-occurrence data. The selected model will contain a soft clustering of words. We can formalize the word clustering problem as that of estimating a soft co-occurrence model.

Numbers of parameters

Table 3.6 shows the numbers of parameters in a hard co-occurrence model and in a soft co-occurrence model. Here N denotes the size of the set of nouns, V the size of the set of verbs, Π_n and Π_v are the partitions in the hard co-occurrence model, and Γ_n and Γ_v are the coverings in the soft co-occurrence model.

Table 3.6: Numbers of parameters in co-occurrence models.

hard co-occurrence model	$O(\lvert\Pi_n\rvert \cdot \lvert\Pi_v\rvert + V + N)$
soft co-occurrence model	$O(\lvert\Gamma_n\rvert \cdot \lvert\Gamma_v\rvert + \sum_{C_n \in \Gamma_n} \lvert C_n\rvert + \sum_{C_v \in \Gamma_v} \lvert C_v\rvert)$

In this thesis, I address only the issue of estimating a hard co-occurrence model.

3.4 Relations between Models

Table 3.7 summarizes the formalization I have made above.

Table 3.7: Summary of the formalization.

Input	Output	Side effect
case slot data	hard/soft case slot model	case slot generalization
case frame data	word/class/slot-based case frame model	case dependency learning
co-occurrence data	hard/soft co-occurrence model	word clustering

The models described above are closely related. The soft case slot model includes the hard case slot model, and the soft co-occurrence model includes the hard co-occurrence model. The slot-based case frame model will become the class-based case frame model when we granulate slot-based case slot values into class-based slot values. The class-based case frame model will become the word-based case frame model when we perform further granulation. The relation between the hard case slot model and the case frame models, that between the hard case slot model and the hard co-occurrence model, and that between the soft case slot model and the soft co-occurrence model are described below.

Hard case slot model and case frame models

The relationship between the hard case slot model and the case frame models may be expressed by transforming the notation of the conditional probability $P(C|v,r)$ in the hard case slot model to

$$P(C|v,r) = P_v(X_r = C|X_r = 1) = \frac{P_v(X_r = C)}{P_v(X_r = 1)}, \tag{3.5}$$

which is the ratio between a marginal probability in the class-based case frame model and a marginal probability in the slot-based case frame model.

This relation (3.5) implies that we can generalize case slots by using the hard case slot model and then acquire class-based case frame patterns by using the class-based case frame model.

Hard case slot model and hard co-occurrence model

If we assume that the verb set consists of a single verb alone, then the hard co-occurrence model *with respect to slot* r becomes

$$P_r(n,v) = P_r(C_n,v) \cdot P_r(n|C_n) \quad n \in C_n.$$

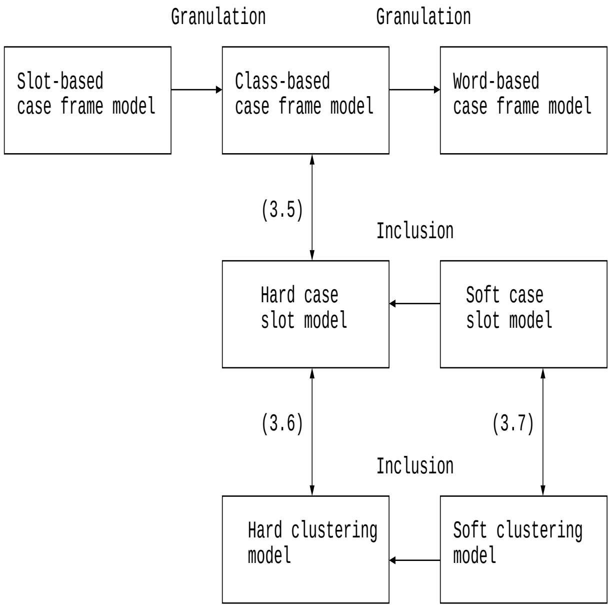


Figure 3.2: Relations between models.

If we further assume that nouns within a same noun class have an equal probability, then we have

$$\frac{P_r(n,v)}{P_r(v)} = P_r(C_n|v) \cdot \frac{1}{|C_n|} \qquad n \in C_n. \tag{3.6}$$

This is no more than the hard case slot model, which has a different notation.

Soft case slot model and soft co-occurrence model

If we assume that the verb set consists of a single verb alone, then the soft co-occurrence model *with respect to slot* r becomes

$$P_r(n,v) = \sum_{C_n \in \Gamma_n} P_r(C_n, v) \cdot P_r(n|C_n).$$

Suppose that $P_r(n|C_n)$ is common to each slot r, then we can denote it as $P(n|C_n)$ and have

$$\frac{P_r(n,v)}{P_r(v)} = \sum_{C_n \in \Gamma_n} P_r(C_n|v) \cdot P(n|C_n). \tag{3.7}$$

This is equivalent to the soft case slot model.

3.5 Discussions

Generation models v.s. decision models

The models defined above are what I call 'generation models.' A case frame generation model is a probability distribution that gives rise to a case frame with a certain probability. In disambiguation, a generation model predicts the *likelihood of the occurrence of each case frame.*

Alternatively, we can define what I call 'decision models' to perform the disambiguation task. A decision model is a conditional distribution which represents the conditional probabilities of disambiguation (or parsing) decisions. For instance, the decision tree model and the decision list model are example decision models (cf., Chapter 2). In disambiguation, a decision model predicts the *likelihood of the correctness of each decision.*

A generation model can generally be represented as a joint distribution $P(\mathbf{X})$ (or a conditional distribution $P(\mathbf{X}|.)$), where random variables $\mathbf{X}$ denote linguistic (syntactical and/or lexical) features. A decision model can generally be represented by a conditional distribution $P(Y|\mathbf{X})$ where random variables $\mathbf{X}$ denote linguistic features and random variable Y denotes usually a *small* number of decisions.

Estimating a generation model requires merely positive examples. On the other hand, estimating a decision model requires both positive and negative examples.

A case frame generation model can be used for purposes other than structural disambiguation. A decision model, on the other hand, is defined specifically for the purpose of disambiguation.

In this thesis, I investigate generation models because of their important generality.

The case slot models are, in fact, 'one-dimensional lexical generation models,' the co-occurrence models are 'two-dimensional lexical generation models,' and the case frame models are 'multi-dimensional lexical generation models.' Note that the case frame models are not simply straightforward extensions of the case slot models and the co-occurrence models; one can easily define different multi-dimensional models as extensions of the case slot models and the co-occurrence models (from one or two dimensions to multi-dimensions).

Linguistic models

The models I have so far defined can also be considered to be linguistic models in the sense that they straightforwardly represent case frame patterns (or selectional patterns, subcategorization patterns) proposed in the linguistic theories of (Fillmore, 1968; Katz and Fodor, 1963; Pollard and Sag, 1987). In other words, they are generally intelligible to humans, because they contain descriptions of language usage.

Probability distributions v.s. probabilistic measures

An alternative to defining probability distributions for lexical knowledge acquisition, and consequently for disambiguation, is to define probabilistic measures (e.g., the association ratio, the selectional association measure). Calculating these measures in a theoretically sound way can be difficult, however, and needs further investigation.

The methods commonly employed to calculate the association ratio measure (cf., Chapter 2) are based on heuristics. For example, it is calculated as

$$\hat{S}(n|v,r) = \log \frac{\hat{P}(n|v,r)}{\hat{P}(n)},$$

where $\hat{P}(n|v,r)$ and $\hat{P}(n)$ denote, respectively, the Laplace estimates of the probabilities $P(n|v,r)$ and $P(n)$. Here, each of the two estimates can only be calculated with a certain degree of precision which depends on the size of training data. Any small inaccuracies in the two may be greatly magnified when they are calculated as a ratio, and this will lead to an extremely unreliable estimate of $S(n|v,r)$ (note that association ratio is an unbounded measure). Since training data is always insufficient, this phenomenon may occur very frequently. Unfortunately, a theoretically sound method of calculation has yet to be developed.

Similarly, a theoretically sound method for calculating the selectional association measure also has yet to be developed. (See (Abe and Li, 1996) for a heuristic method for learning a similar measure on the basis of the MDL principle.) In this thesis I employ probability distributions rather than probabilistic measures.

3.6 Disambiguation Methods

The models proposed above can be independently used for disambiguation purposes, they can also be combined into a single natural language analysis system. In this section, I first describe how they can be independently used and then how they can be combined.

Using case frame models

Suppose for example that in the analysis of the sentence

The girl will fly a jet from Tokyo,

the following alternative interpretations are obtained.

(fly (arg1 girl) (arg2 (jet)) (from Tokyo))

(fly (arg1 girl) (arg2 (jet (from Tokyo)))).

We wish to select the more appropriate of the two interpretations. Suppose for simplicity that there are four possible case slots for the verb 'fly,' and there is only one possible case slot for the noun 'jet.' A disambiguation method based on word-based case frame models would calculate the following likelihood values and select the interpretation with higher likelihood value:

$$P_{\text{fly}}(X_{\text{arg1}} = \text{girl}, X_{\text{arg2}} = \text{jet}, X_{\text{from}} = \text{Tokyo}, X_{\text{to}} = 0) \cdot P_{\text{jet}}(X_{\text{from}} = 0)$$

and

$$P_{\text{fly}}(X_{\text{arg1}} = \text{girl}, X_{\text{arg2}} = \text{jet}, X_{\text{from}} = 0, X_{\text{to}} = 0) \cdot P_{\text{jet}}(X_{\text{from}} = \text{Tokyo}).$$

If the former is larger than the latter, we select the former interpretation, otherwise we select the latter interpretation.

If we assume here that case slots are independent, then we need only compare

$$P_{\text{fly}}(X_{\text{from}} = \text{Tokyo}) \cdot P_{\text{jet}}(X_{\text{from}} = 0)$$

and

$$P_{\text{fly}}(X_{\text{from}} = 0) \cdot P_{\text{jet}}(X_{\text{from}} = \text{Tokyo}).$$

Similarly, when the models are slot-based and the case slots are assumed to be independent, we need only compare

$$\begin{aligned} & P_{\text{fly}}(X_{\text{from}} = 1) \cdot P_{\text{jet}}(X_{\text{from}} = 0) \\ & = P_{\text{fly}}(X_{\text{from}} = 1) \cdot \Big(1 - P_{\text{jet}}(X_{\text{from}} = 1)\Big) \end{aligned}$$

and

$$\begin{aligned} & P_{\text{fly}}(X_{\text{from}} = 0) \cdot P_{\text{jet}}(X_{\text{from}} = 1) \\ & = \Big(1 - P_{\text{fly}}(X_{\text{from}} = 1)\Big) \cdot P_{\text{jet}}(X_{\text{from}} = 1). \end{aligned}$$

That is to say, we need noly compare

$$P_{\text{fly}}(X_{\text{from}} = 1)$$

and

$$P_{\text{jet}}(X_{\text{from}} = 1).$$

The method proposed by Hindle and Rooth (1991) in fact compares the same probabilities; they do it by means of statistical hypothesis testing.

Using hard case slot models

Another way of conducting disambiguation under the assumption that case slots are independent is to employ the hard case slot model. Specifically we compare

$$P(\text{Tokyo}|\text{fly}, \text{from})$$

and

$$P(\text{Tokyo}|\text{jet}, \text{from}).$$

If the former is larger than the latter, we select the former interpretation, otherwise we select the latter interpretation.

Using hard co-occurrence models

We can also use the hard co-occurrence model to perform the disambiguation task, under the assumption that case slots are independent. Specifically, we compare

$$P_{\text{from}}(\text{Tokyo}|\text{fly}) = \frac{P_{\text{from}}(\text{Tokyo}, \text{fly})}{\sum_{n \in \mathcal{N}} P_{\text{from}}(n, \text{fly})}$$

and

$$P_{\text{from}}(\text{Tokyo}|\text{jet}) = \frac{P_{\text{from}}(\text{Tokyo}, \text{jet})}{\sum_{n \in \mathcal{N}} P_{\text{from}}(n, \text{jet})}.$$

Here, $P_{\text{from}}(\text{Tokyo}, \text{fly})$ is calculated on the basis of a hard co-occurrence model over the set of nouns and the set of verbs with respect to the 'from' slot, and $P_{\text{from}}(\text{Tokyo}, \text{jet})$ on the basis of a hard co-occurrence model over the set of nouns with respect to the 'from' slot.

Since the joint probabilities above are all estimated on the basis of class-based models, the conditional probabilities are in fact calculated on the basis of not only the co-occurrences of the related words but also of those of similar words. That means that this disambiguation method is similar to the similarity-based approach (cf., Chapter 2). The difference is that the method described here is based on a probability model, while the similarity-based approach usually is based on heuristics.

A combined method

Let us next consider a method based on combination of the above models.

We first employ the hard co-occurrence model to construct a thesaurus for each case slot (we can, however, construct only thesauruses for which there are enough co-occurrence data with respect to the corresponding case slots). We next employ the hard case slot model to generalize values of case slots into word classes (word classes used in a hard case slot model can be either from a hand-made thesaurus or from an automatically constructed thesaurus; cf., Chapter 4). Finally, we employ the class-based case frame model to learn class-based case frame patterns.

In disambiguation, we refer to the case frame patterns, calculate likelihood values for the ambiguous case frames, and select the most likely case frame as output.

With regard to the above example, we can calculate and compare the following likelihood values:

$$L(1) = P_{\text{fly}}(X_{\text{arg1}} = \langle\text{person}\rangle, X_{\text{arg2}} = \langle\text{airplane}\rangle, X_{\text{from}} = \langle\text{place}\rangle) \cdot P_{\text{jet}}(X_{\text{from}} = 0)$$

and

$$L(2) = P_{\text{fly}}(X_{\text{arg1}} = \langle\text{person}\rangle, X_{\text{arg2}} = \langle\text{airplane}\rangle, X_{\text{from}} = 0) \cdot P_{\text{jet}}(X_{\text{from}} = \langle\text{place}\rangle),$$

assuming that there are only three case slots: arg1, arg2 and 'from' for the verb 'fly,' and there is one case slot: 'from' for the noun 'jet.' Here $\langle\cdots\rangle$ denotes a word class. We make the pp-attachment decision as follows: if $L(1) > L(2)$, we attach the phrase 'from Tokyo' to 'fly;' if $L(1) < L(2)$, we attach it to 'jet;' otherwise we make no decision.

Unfortunately, it is still difficult to attain high performance with this method at the current stage of statistical language processing, since the corpus data currently available is far less than that necessary to estimate accurately the class-based case frame models.

3.7 Summary

I have proposed the soft/hard case slot model for case slot generalization, the word-based/class-based/slot-based case frame model for case dependency learning, and the soft/hard co-occurrence model for word clustering. In Chapter 4, I will describe a method for learning the hard case slot model, i.e., generalizing case slots; in Chapter 5, a method for learning the case frame model, i.e., learning case dependencies; and in Chapter 6, a method for learning the hard co-occurrence model, i.e., conducting word clustering. In Chapter 7, I will describe a disambiguation method, which is based on the learning methods proposed in Chapters 4 and 6. (See Figure 1.1.)

Chapter 4

Case Slot Generalization

Make everything as simple as possible - but not simpler.

- Albert Einstein

In this chapter, I describe one method for learning the hard case slot model, i.e., generalizing case slots.

4.1 Tree Cut Model

As described in Chapter 3, we can formalize the case slot generalization problem into that of estimating a conditional probability distribution referred to as a 'hard case slot model.' The problem thus turns to be that of selecting the best model from among all possible hard case slot models. Since the number of partitions for a set of nouns is very large, the number of such models is very large, too. The problem of estimating a hard case slot model, therefore, is most likely intractable. (The number of partitions for a set of nouns is $\sum_{i=1}^{N}\sum_{j=1}^{i}\frac{(-1)^{i-j}\cdot j^{N}}{(i-j)!\cdot j!}$, where N is the size of the set of nouns (cf., (Knuth, 1973)), and is roughly of order $O(N^N)$.)

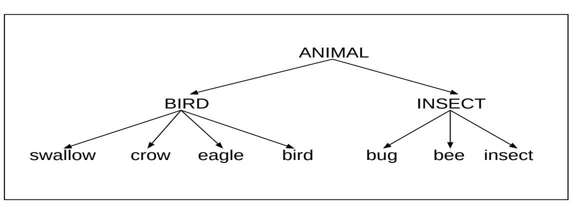


Figure 4.1: An example thesaurus.

To deal with this difficulty, I take the approach of restricting the class of case slot models. I reduce the number of partitions necessary for consideration by using a thesaurus, following a similar proposal given in (Resnik, 1992). Specifically, I restrict attention to those partitions that exist within the thesaurus in the form of a cut. Here by 'thesaurus' is meant a rooted tree in which each leaf node stands for a noun, while each internal node represents a noun class, and a directed link represents set inclusion (cf., Figure 4.1). A 'cut' in a tree is any set of nodes in the tree that can represent a partition of the given set of nouns. For example, in the thesaurus of Figure 4.1, there are five cuts: [ANIMAL],[BIRD, INSECT], [BIRD, bug, bee, insect], [swallow, crow, eagle, bird, INSECT], and [swallow, crow, eagle, bird, bug, bee, insect].

The class of 'tree cut models' *with respect to a fixed thesaurus tree* is then obtained by restricting the partitions in the definition of a hard case slot model to be those that are present as a cut in that thesaurus tree. The number of models, then, is drastically reduced, and is of order $\Theta(2^{\frac{N}{b}})$ when the thesaurus tree is a complete b-ary tree, because the number of cuts in a complete b-ary tree is of that order (see Appendix A.3). Here, N denotes the number of leaf nodes, i.e., the size of the set of nouns.

A tree cut model M can be represented by a pair consisting of a tree cut Γ (i.e., a discrete model), and a probability parameter vector θ of the same length, that is,

$$M = (\Gamma, \theta),$$

where Γ and θ are

$$\Gamma = [C_1, C_2, \cdots, C_{k+1}], \theta = [P(C_1), P(C_2), \cdots, P(C_{k+1})],$$

where $C_1, C_2, \cdots, C_{k+1}$ forms a cut in the thesaurus tree and where $\sum_{i=1}^{k+1} P(C_i) = 1$ is satisfied. Hereafter, for simplicity I sometimes write $P(C_i)$ for $P(C_i|v, r)$, where $i = 1, \cdots, (k+1)$.

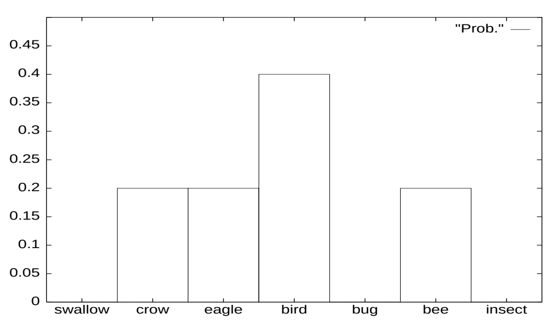


Figure 4.2: A tree cut model with [swallow, crow, eagle, bird, bug, bee, insect].

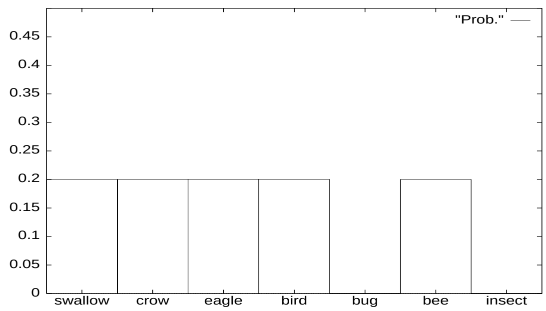


Figure 4.3: A tree cut model with [BIRD, bug, bee, insect].

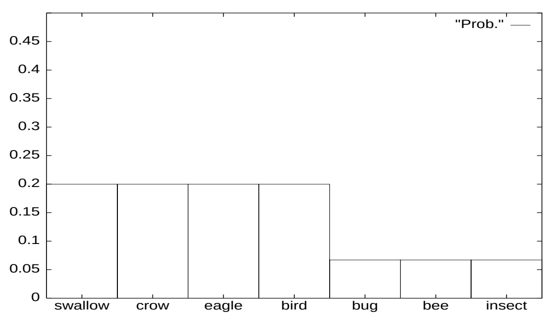


Figure 4.4: A tree cut model with [BIRD, INSECT].

If we employ MLE for parameter estimation, we can obtain five tree cut models from the case slot data in Figure 2.1; Figures 4.2-4.4 show three of these. For example, $\hat{M} = ([\mathrm{BIRD}, \mathrm{bug}, \mathrm{bee}, \mathrm{insect}], [0.8, 0, 0.2, 0])$ shown in Figure 4.3 is one such tree cut model. Recall that $\hat{M}$ defines a conditional probability distribution $P_{\hat{M}}(n|v,r)$ in the following way: for any noun that is in the tree cut, such as 'bee,' the probability is given as explicitly specified by the model, i.e., $P_{\hat{M}}(\mathrm{bee}|\mathrm{fly}, \mathrm{arg1}) = 0.2$; for any class in the tree cut, the probability is distributed uniformly to all nouns included in it. For example, since there are four nouns that fall under the class BIRD, and 'swallow' is one of them, the probability of 'swallow' is thus given by $P_{\hat{M}}(\mathrm{swallow}|\mathrm{fly}, \mathrm{arg1}) = 0.8/4 = 0.2$. Note that the probabilities assigned to the nouns under BIRD are *smoothed*, even if the nouns have different observed frequencies.

In this way, the problem of generalizing the values of a case slot has been formalized into that of estimating a model from the class of tree cut models for some fixed thesaurus tree.

4.2 MDL as Strategy

The question now becomes what strategy (criterion) we should employ to select the best tree cut model. I propose to adopt the MDL principle.

Table 4.1: Number of parameters and KL divergence for the five tree cut models.

Γ	Number of parameters	KL divergence
[ANIMAL]	0	1.4
[BIRD, INSECT]	1	0.72
[BIRD, bug, bee, insect]	3	0.4
[swallow, crow, eagle, bird, INSECT]	4	0.32
[swallow, crow, eagle, bird, bug, bee, insect]	6	0

In our current problem, a model nearer the root of the thesaurus tree, such as that of Figure 4.4, generally tends to be simpler (in terms of the number of parameters), but also tends to have a poorer fit to the data. By way of contrast, a model nearer the leaves of the thesaurus tree, such as that in Figure 4.2, tends to be more complex, but also tends to have a better fit to the data. Table 4.1 shows the number of free parameters and the 'KL divergence' between the empirical distribution (namely, the word-based distribution estimated by MLE) of the data shown in Figure 2.2 and each of the five tree cut models.[1] In the table, we can see that there is a trade-off between the simplicity of a model and the goodness of its fit to the data. The use of MDL can balance the trade-off relationship.

Let us consider how to calculate description length for the current problem, where the notations are slightly different from those in Chapter 2. Suppose that S denotes a sample (or data), which is a *multi-set* of examples, each of which is an *occurrence* of a noun at a slot r for a verb v (i.e., duplication is allowed). Further suppose that $|S|$ denotes the size of S, and $n \in S$ indicates the inclusion of n in S. For example, the column labeled 'slot value' in Table 2.3 represents a sample S for the arg1 slot for 'fly,' and in this case $|S| = 10$.

Given a sample S and a tree cut Γ, we can employ MLE to estimate the parameters of the corresponding tree cut model $\hat{M} = (\Gamma, \hat{\theta})$, where $\hat{\theta}$ denotes the estimated parameters.

The total description length $l(\hat{M}, S)$ of the tree cut model $\hat{M}$ and the data S observed through $\hat{M}$ may be computed as the sum of model description length $l(\Gamma)$,

[1]The KL divergence (also known as 'relative entropy') is a measure of the 'distance' between two probability distributions, and is defined as $D(P||Q) = \sum_i p_i \cdot \log \frac{p_i}{q_i}$ where p_i and q_i represent, respectively, probabilities in discrete distributions P and Q (Cover and Thomas, 1991).

parameter description length $l(\hat{\theta}|\Gamma)$, and data description length $l(S|\Gamma,\hat{\theta})$, i.e.,

$$l(\hat{M},S) = l((\Gamma,\hat{\theta}),S) = l(\Gamma) + l(\hat{\theta}|\Gamma) + l(S|\Gamma,\hat{\theta}).$$

Model description length $l(\Gamma)$, here, may be calculated as[2]

$$l(\Gamma) = \log|\mathcal{G}|,$$

where $\mathcal{G}$ denotes the set of all cuts in the thesaurus tree T. From the viewpoint of Bayesian Estimation, this corresponds to assuming that each tree cut model to be equally likely *a priori*.

Parameter description length $l(\hat{\theta}|\Gamma)$ may be calculated by

$$l(\hat{\theta}|\Gamma) = \frac{k}{2}\cdot\log|S|,$$

where $|S|$ denotes the sample size and k denotes the number of free parameters in the tree cut model, i.e., k equals the number of nodes in Γ minus one.

Finally, data description length $l(S|\Gamma,\hat{\theta})$ may be calculated as

$$l(S|\Gamma,\hat{\theta}) = -\sum_{n\in S}\log\hat{P}(n),$$

where for simplicity I write $\hat{P}(n)$ for $P_{\hat{M}}(n|v,r)$. Recall that $\hat{P}(n)$ is obtained by MLE, i.e.,

$$\hat{P}(n) = \frac{1}{|C|}\cdot\hat{P}(C)$$

for each $n \in C$, where for each $C \in \Gamma$

$$\hat{P}(C) = \frac{f(C)}{|S|},$$

where $f(C)$ denotes the frequency of nouns in class C in data S.

With the description length defined in the above manner, we wish to select a model with the minimum description length, and then output it as the result of generalization. Since every tree cut has an equal $l(\Gamma)$, technically we need only calculate and compare $L'(\hat{M},S) = l(\hat{\theta}|\Gamma) + l(S|\Gamma,\hat{\theta})$. In the discussion which follows, I sometimes use $L'(\Gamma)$ for $L'(\hat{M},S)$, where Γ is the tree cut of $\hat{M}$, for the sake of simplicity.

The description lengths of the data in Figure 2.1 for the tree cut models with respect to the thesaurus tree in Figure 4.1 are shown in Table 4.3. (Table 4.2 shows how the description length is calculated for the model with tree cut [BIRD, bug, bee, insect].) These figures indicate that according to MDL, the model in Figure 4.4 is the best model. Thus, given the data in Table 2.3 as input, we are able to obtain the generalization result shown in Table 4.4.

[2]Throughout this thesis, 'log' denotes the logarithm to base 2.

Table 4.2: Calculating description length.

C	BIRD	bug	bee	insect
$f(C)$	8	0	2	0
$\lvert C\rvert$	4	1	1	1
$\hat{P}(C)$	0.8	0.0	0.2	0.0
$\hat{P}(n)$	0.2	0.0	0.2	0.0
Γ	[BIRD, bug, bee, insect]			
$l(\hat{\theta}\lvert\Gamma)$	$\frac{(4-1)}{2} \times \log 10 = 4.98$			
$l(S\lvert\Gamma, \hat{\theta})$	$-(2+4+2+2) \times \log 0.2 = 23.22$			

Table 4.3: Description lengths for the five tree cut models.

Γ	$l(\hat{\theta}\lvert\Gamma)$	$l(S\lvert\Gamma, \hat{\theta})$	$L'(\Gamma)$
[ANIMAL]	0	28.07	28.07
[BIRD, INSECT]	1.66	26.39	<u>28.05</u>
[BIRD, bug, bee, insect]	4.98	23.22	28.20
[swallow, crow, eagle, bird, INSECT]	6.64	22.39	29.03
[swallow, crow, eagle, bird, bug, bee, insect]	9.97	19.22	29.19

Let us next consider some justifications for calculating description lengths in the above ways.

For the model description length $l(\Gamma)$, I assumed the length to be equal for all the *discrete* tree cut models. We could, alternatively, have assigned larger code lengths to models nearer the root node and smaller code lengths to models nearer the leaf nodes. I chose not to do so for the following reasons: (1) in general, when we have no information about a class of models, it is optimal to assume, on the basis of the 'minmax strategy' in Bayesian Estimation, that each model has equal prior probability (i.e., to assume 'equal ignorance'); (2) when the data size is large enough, the model description length, which is only of order $O(1)$, will be negligible compared to the parameter description length, which is of order $O(\log|S|)$; (3) this way of calculating the model description length is

Table 4.4: Generalization result.

Verb	Slot name	Slot value	Probability
fly	arg1	BIRD	0.8
fly	arg1	INSECT	0.2

compatible with the dynamic-programming-based learning algorithm described below.

With regard to the calculation of parameter description length $l(\hat{\theta}|\Gamma)$, we should note that the use of the looser form (2.9) rather than the more precise form (2.14) is done out of similar consideration of compatibility with the dynamic programming technique.

4.3 Algorithm

In generalizing the values of a case slot using MDL, if computation time were of no concern, one could in principle calculate the description length for every possible tree cut model and output a model with the minimum description length as a generalization result, But since the number of cuts in a thesaurus tree is usually exponential (cf., Appendix A.3), it is impractical to do so. Nonetheless, we were able to devise a simple and efficient algorithm, based on dynamic programming, which is guaranteed to find a model with the minimum description length.

The algorithm, which we call 'Find-MDL,' recursively finds the optimal submodel for each child subtree of a given (sub)tree and follows one of two possible courses of action: (1) it either combines these optimal submodels and returns this combination as output, or (2) it collapses all these optimal submodels into the (sub)model containing the root node of the given (sub)tree. Find-MDL simply chooses the course of action which will result in the shorter description length (cf., Figure 4.5). Note that for simplicity I describe Find-MDL as outputting a tree cut, rather than a tree cut model.

Note in the above algorithm that the parameter description length is calculated as $\frac{k+1}{2} \cdot \log|S|$, where $k+1$ is the number of nodes in the current cut, both when t is the entire tree and when it is a proper subtree. This contrasts with the fact that the number of *free* parameters is k for the former, while it is $k+1$ for the latter. For the purpose of finding a tree cut model with the minimum description length, however, this distinction can be ignored (cf., Appendix A.4).

Figure 4.6 illustrates how the algorithm works. In the recursive application of Find-MDL on the subtree rooted at AIRPLANE, the if-clause on line 9 is true since $L'([\text{AIRPLANE}]) = 32.20$, $L'([\text{jet}, \text{helicopter}, \text{airplane}]) = 32.72$, and hence [AIRPLANE] is returned. Similarly, in the application of Find-MDL on the subtree rooted at ARTIFACT, the same if-clause is false since $L'([\text{VEHICLE}, \text{AIRPLANE}]) = 40.83$, $L'([\text{ARTIFACT}]) = 40.95$, and hence [VEHICLE, AIRPLANE] is returned.

Concerning the above algorithm, the following proposition holds:

Proposition 1 *The algorithm* Find-MDL *terminates in time $O(N)$, where N denotes the number of leaf nodes in the thesaurus tree T, and it outputs a tree cut model of T with the minimum description length (with respect to the coding scheme described in Section 4.2).*

See Appendix A.4 for a proof of the proposition.

Let t denote a thesaurus (sub)tree, while root(t) denotes the root of t.
Let c denote a tree cut in t. Initially t is set to the entire tree.

```
algorithm Find-MDL(t):= c
if
    t is a leaf node
then
    return([t]);
else
    For each child subtree t_i of t c_i :=Find-MDL(t_i);
    c:= append(c_i);
    if
        L'([root(t)]) < L'(c)
    then
        return([root(t)]);
    else
        return(c).
```

Figure 4.5: The Find-MDL algorithm.

4.4 Advantages

Coping with the data sparseness problem

Using the MDL-based method described above, we can generalize the values of a case slot. The probability of a noun being the value of a slot can then be represented as a conditional probability estimated (smoothed) from a class-based model on the basis of the MDL principle.

The advantage of this method over the word-based method described in Chapter 2 lies in its ability to cope with the data sparseness problem. Formalizing this problem as a statistical estimation problem that includes model selection enables us to select models with various complexities, while employing MDL enables us to select, on the basis of training data, a model with the most appropriate level of complexity.

Generalization

The case slot generalization problem can also be restricted to that of generalizing individual nouns present in case slot data into classes of nouns present in a given thesaurus. For example, given the thesaurus in Figure 4.1 and frequency data in Figure 2.1, we would like our system to judge that the class 'BIRD' and the noun 'bee' can be the value of the arg1 slot for the verb 'fly.' The problem of deciding whether to stop generalizing at 'BIRD' and 'bee' or to continue generalizing further to 'ANIMAL'

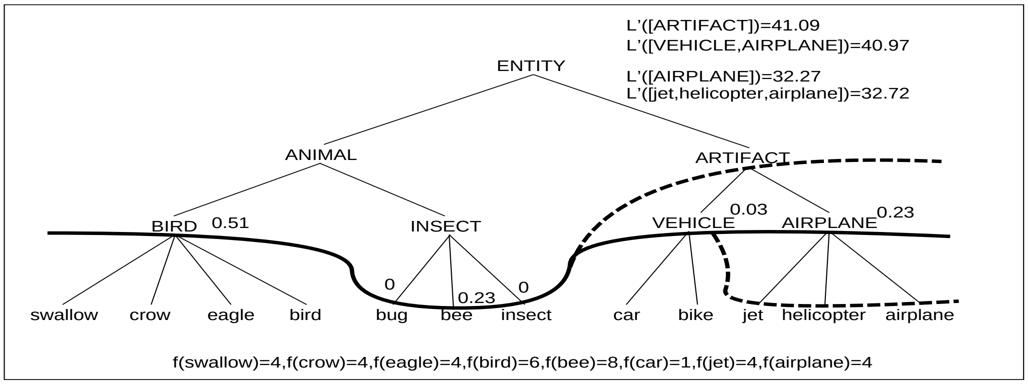


Figure 4.6: An example application of Find-MDL.

has been addressed by a number of researchers (cf., (Webster and Marcus, 1989; Velardi, Pazienza, and Fasolo, 1991; Nomiyama, 1992)). The MDL-based method described above provides a disciplined way to realize this on the basis of data compression and statistical estimation.

The MDL-based method, in fact, conducts generalization in the following way. When the differences between the frequencies of the words in a class are not large enough (relative to the entire data size and the number of the words), it generalizes them into the class. When the differences are especially noticeable (relative to the entire data size and the number of the words), on the other hand, it stops generalization at that level.

As described in Chapter 3, the class of hard case slot models contains all of the possible models for generalization, if we view the generalization process as that of finding the best configuration of words such that the words in each class are equally likely to the value of a case slot. And thus if we could estimate the best model from the class of hard case slot models on the basis of MDL, we would be able to obtain the most appropriate generalization result. When we make use of a thesaurus (hand-made or automatically constructed) to restrict the model class, the generalization result will inevitablely be affected by the thesaurus used, and the tree cut model selected may be a loose approximation of the best model. Because MDL achieves a balanced trade-off between model simplicity and data fit, we may expect that the model it selects will represent a reasonable compromise.

Coping with extraction noise

Avoiding the influence of noise in case slot data is another problem that needs consideration in case slot generalization. For example, suppose that the case slot data

on the noun 'car' in Figure 4.6 is noise. In such case, the MDL-based method tends to generalize a noun to a class at quite high a level, since the differences between the frequency of the noun and those of its neighbors are not high (e.g., $f(\text{car}) = 1$ and $f(\text{bike}) = 0$). The probabilities of the generalized classes will, however, be small. If we discard those classes in the obtained tree cut that have small probabilities, we will still acquire reliable generalization results. That is to say, the proposed method is robust against noise.

4.5 Experimental Results

4.5.1 Experiment 1: qualitative evaluation

I have applied the MDL-based generalization method to a data corpus and inspected the obtained tree cut models to see if they agree with human intuition. In the experiments, I used existing techniques (cf., (Manning, 1992; Smadja, 1993)) to extract case slot data from the *tagged* texts of the Wall Street Journal corpus (ACL/DCI CD-ROM1) consisting of 126,084 sentences. I then applied the method to generalize the slot values.

Table 4.5 shows some example case slot data for the arg2 slot for the verb 'eat.' There were some extraction errors present in the data, but I chose not to remove them because extraction errors are such a generally common occurrence that a realistic evaluation should include them.

Table 4.5: Example input data (for the arg2 slot for 'eat').

eat arg2 food	3	eat arg2 lobster	1	eat arg2 seed	1
eat arg2 heart	2	eat arg2 liver	1	eat arg2 plant	1
eat arg2 sandwich	2	eat arg2 crab	1	eat arg2 elephant	1
eat arg2 meal	2	eat arg2 rope	1	eat arg2 seafood	1
eat arg2 amount	2	eat arg2 horse	1	eat arg2 mushroom	1
eat arg2 night	2	eat arg2 bug	1	eat arg2 ketchup	1
eat arg2 lunch	2	eat arg2 bowl	1	eat arg2 sawdust	1
eat arg2 snack	2	eat arg2 month	1	eat arg2 egg	1
eat arg2 jam	2	eat arg2 effect	1	eat arg2 sprout	1
eat arg2 diet	1	eat arg2 debt	1	eat arg2 nail	1
eat arg2 pizza	1	eat arg2 oyster	1		

When generalizing, I used the noun taxonomy of WordNet (version1.4) (Miller, 1995) as the thesaurus. The noun taxonomy of WordNet is structured as a directed acyclic graph (DAG), and each of its nodes stands for a word sense (a concept), often containing several words having the same word sense. WordNet thus deviates from the notion of a thesaurus as defined in Section 4.1 – a tree in which each leaf node stands

for a noun, and each internal node stands for a class of nouns; we need to take a few measures to deal with this.

First, each subgraph having multiple parents is copied so that the WordNet is transformed into a tree structure [3] and the algorithm Find-MDL can be applied. Next, the issue of word sense ambiguity is heuristically addressed by equally dividing the observed frequency of a noun between all the nodes containing that noun. Finally, the highest nodes actually containing the values of the slot are used to form the 'staring cut' from which to begin generalization and the frequencies of all the nodes below to a node in the starting cut are added to that node. Since word senses of nouns that occur in natural language tend to concentrate in the middle of a taxonomy,[4] a starting cut given by this method usually falls around the middle of the thesaurus.

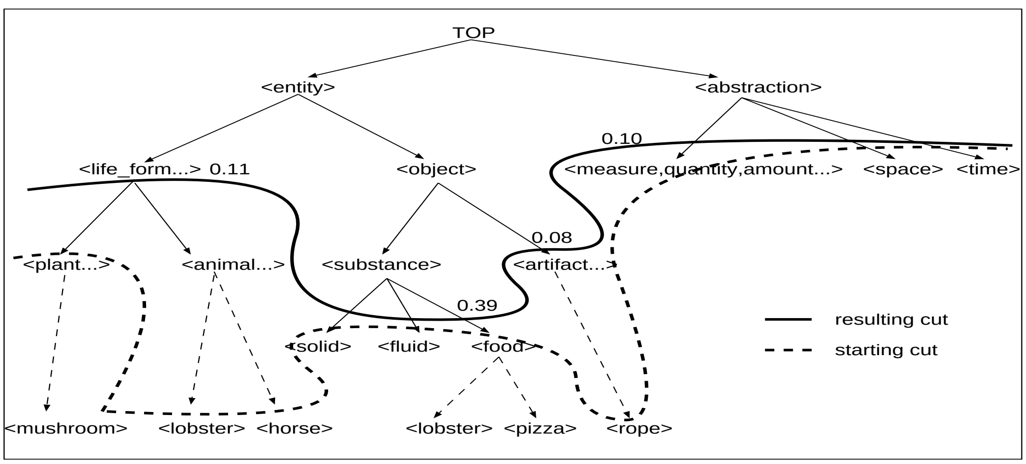


Figure 4.7: Example generalization result (for the arg2 slot for 'eat').

Figure 4.7 indicates the starting cut and the resulting cut in WordNet for the arg2 slot for 'eat' with respect to the data in Table 4.5, where $\langle \cdots \rangle$ denotes a node in WordNet. The starting cut consists of those nodes $\langle$plant, $\cdots\rangle$,$\langle$food$\rangle$,etc. which are the highest nodes containing the values of the arg2 slot for 'eat.' Since $\langle$food$\rangle$ has significantly more frequencies than its neighbors $\langle$solid$\rangle$ and $\langle$fluid$\rangle$, MDL has the generalization stop there. By way of contrast, because the nodes under $\langle$life_form, $\cdots\rangle$ have relatively small differences in their frequencies, they are generalized to the node $\langle$life_form, $\cdots\rangle$.

[3]In fact, there are only few nodes in WordNet, which have multiple parent nodes, i.e., the structure of WordNet approximates that of a tree.

[4]Cognitive scientists have observed that concepts in the middle of a taxonomy tend to be more important with respect to learning, recognition, and memory, and their linguistic expressions occur more frequently in natural language – a phenomenon known as 'basic level primacy.' (cf., (Lakoff, 1987))

The same is true of the nodes under ⟨artifact, ···⟩. Since ⟨···, amount, ···⟩ has a much higher frequency than its neighbors ⟨time⟩ and ⟨space⟩, generalization does not proceed any higher. All of these results seem to agree with human intuition, indicating that the method results in an appropriate level of generalization.

Table 4.6 shows generalization results for the arg2 slot for 'eat' and three other arbitrarily selected verbs, where classes are sorted in descending order with respect to probability values. (Classes with probabilities less than 0.05 have been discarded due to space limitations.) Despite the fact that the employed extraction method is not noise-free, and word sense ambiguities remain after extraction, the generalization results seem to agree with intuition to a satisfactory degree. (With regard to noise, at least, this is not too surprising since the noisy portion usually has a small probability and thus tends to be discarded.)

Table 4.6: Examples of generalization results.

Class	Probability	Example words
arg2 slot of 'eat'		
⟨food, nutrient⟩	0.39	pizza, egg
⟨life_form, organism, ···⟩	0.11	lobster, horse
⟨measure, quantity, ···⟩	0.10	amount *of*
⟨artifact, article, ···⟩	0.08	*as if eat* rope
arg2 slot of 'buy'		
⟨object, ···⟩	0.30	computer, painting
⟨asset⟩	0.10	stock, share
⟨group, grouping⟩	0.07	company, bank
⟨legal_document, ···⟩	0.05	security, ticket
arg2 slot of 'fly'		
⟨entity⟩	0.35	airplane, flag, executive
⟨linear_measure, ···⟩	0.28	mile
⟨group, grouping⟩	0.08	delegation
arg2 slot of 'operate'		
⟨group, grouping⟩	0.13	company, fleet
⟨act, human_action, ···⟩	0.13	flight, operation
⟨structure, ···⟩	0.12	center
⟨abstraction⟩	0.11	service, unit
⟨possession⟩	0.06	profit, earnings

Table 4.7 shows the computation time required (on a SPARC 'Ultra 1' work station, not including that for loading WordNet) to obtain the results shown in Table 4.6. Even though the noun taxonomy of WordNet is a large thesaurus containing approximately 50,000 nodes, the MDL-based method still manages to generalize case slots efficiently

Table 4.7: Required computation time and number of generalized levels.

Verb	CPU time (second)	Average number of generalized levels
eat	1.00	5.2
buy	0.66	4.6
fly	1.11	6.0
operate	0.90	5.0
Average	0.92	5.2

with it. The table also shows the average number of levels generalized for each slot, i.e., the average number of links between a node in the starting cut and its ancestor node in the resulting cut. (For example, the number of levels generalized for $\langle \text{plant}, \cdots \rangle$ is one in Figure 4.7.) One can see that a significant amount of generalization is performed by the method – the resulting tree cut is on average about 5 levels higher than the starting cut.

4.5.2 Experiment 2: pp-attachment disambiguation

Case slot patterns obtained by the method can be used in various tasks in natural language processing. Here, I test the effectiveness of the use of the patterns in pp-attachment disambiguation.

In the experiments described below, I compare the performance of the proposed method, referred to as 'MDL,' against the methods proposed by (Hindle and Rooth, 1991), (Resnik, 1993b), and (Brill and Resnik, 1994), referred to respectively as 'LA,' 'SA,' and 'TEL.'

Data set

As a data set, I used the bracketed data of the Wall Street Journal corpus (Penn Tree Bank 1) (Marcus, Santorini, and Marcinkiewicz, 1993). First I randomly selected one of the 26 directories of the WSJ files as test data and what remained as training data. I repeated this process ten times and obtained ten sets of data consisting of different training and test data. I used these ten data sets to conduct *cross validation*, as described below.

From the *test* data in each data set, I extracted (v, n_1, p, n_2) quadruples using the extraction tool provided by the Penn Tree Bank called 'tgrep.' At the same time, I obtained the *answer* for the pp-attachment for each quadruple. I did not double-check to confirm whether or not the answers were actually correct. From the *training* data of each data set, I then extracted (v, p) and (n_1, p) doubles, and (v, p, n_2) and (n_1, p, n_2) triples using tools I developed. I also extracted quadruples from the training data as before. I then applied 12 heuristic rules to further preprocess the data; this

processing included (1) changing the inflected form of a word to its stem form, (2) replacing numerals with the word 'number,' (3) replacing integers between 1900 and 2999 with the word 'year,' (4) replacing 'co.,' 'ltd.' with the word 'company,' (5) etc. After preprocessing some minor errors still remained, but I did not attempt to remove them because of lacking a good method to do so automatically. Table 4.8 shows the number of different types of data obtained in the above process.

Table 4.8: Number of data items.

Training data	
average number of doubles per data set	91218.1
average number of triples per data set	91218.1
average number of quadruples per data set	21656.6
Test data	
average number of quadruples per data set	820.4

Experimental procedure

I first compared the accuracy and coverage for MDL, SA and LA.

For MDL, n_2 is generalized on the basis of two sets of triples (v, p, n_2) and (n_1, p, n_2) that are given as training data for each data set, with WordNet being used as the thesaurus in the same manner as it was in Experiment 1. When disambiguating, rather than comparing $\hat{P}(n_2|v,p)$ and $\hat{P}(n_2|n_1,p)$ I compare $\hat{P}(C_1|v,p)$ and $\hat{P}(C_2|n_1,p)$, where C_1 and C_2 are classes in the output tree cut models dominating n_2[5]; because I empirically found that to do so gives a slightly better result. For SA, I employ a basic application (also using WordNet) in which n_2 is generalized given (v, p, n_2) and (n_1, p, n_2) triples. For disambiguation I compare $\hat{A}(n_2|v,p)$ and $\hat{A}(n_2|n_1,p)$ (defined in (2.2) in Chapter 2)). For LA, I estimate $\hat{P}(p|v)$ and $\hat{P}(p|n_1)$ from the training data of each data set and compare them for disambiguation.

I then evaluated the results achieved by the three methods in terms of accuracy and coverage. Here 'coverage' refers to the percentage of test data by which a disambiguation method can reach a decision, and 'accuracy' refers to the proportion of correct decisions among all decisions made.

Figure 4.8 shows the accuracy-coverage curves for the three methods. In plotting these curves, I first compare the respective values for the two possible attachments. If the difference between the two values exceeds a certain threshold, I make the decision to attach at the higher-value site. The threshold here was set successively to 0,0.01,0.02,0.05,0.1,0.2,0.5,and 0.75 for each of the three methods. When the difference

[5] Recall that a node in WordNet represents a word sense and not a word, n_2 can belong to several different classes in the thesaurus. In fact, I compared $\max_{C_i \ni n_2}(\hat{P}(C_i|v,p))$ and $\max_{C_j \ni n_2}(\hat{P}(C_j|n_1,p))$.

between the two values is less than the threshold, no decision is made. These curves were obtained by averaging over the ten data sets. Figure 4.8 shows that, with respect to accuracy-coverage curves, MDL outperforms both SA and LA throughout, while SA is better than LA.

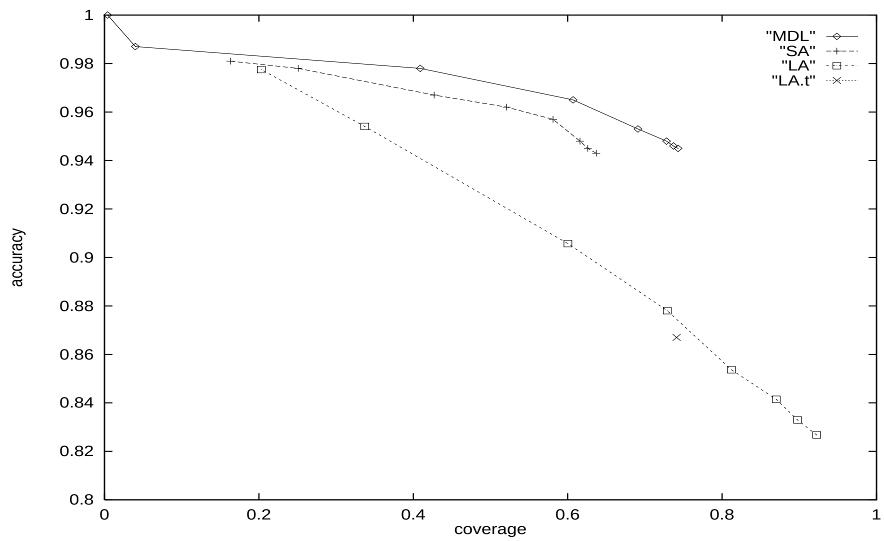


Figure 4.8: Accuracy-coverage plots for MDL, SA, and LA.

I also implemented the method proposed by (Hindle and Rooth, 1991) which makes disambiguation judgements using t-scores (cf., Chapter 2). Figure 4.8 shows the result as 'LA.t,' where the threshold for the t-score is 1.28 (at a significance level of 90 percent.)

Next, I tested the method of applying a default rule after applying each method. That is, attaching (p, n_2) to v for the part of the test data for which no decision was made by the method in question. (Interestingly, over the data set as a whole it is more favorable to attach (p, n_2) to n_1, but for what remains after applying LA, SA, and MDL, it turns out to be more favorable to attach (p, n_2) to v.) I refer to these combined methods as MDL+Default, SA+Default, LA+Default, and LA.t+Default. Table 4.9 shows the results, again averaged over the ten data sets.

Finally, I used transformation-based error-driven learning (TEL) to acquire transformation rules for each data set and applied the obtained rules to disambiguate the test data (cf., Chapter 2). The average number of obtained rules for a data set was 2752.3. Table 4.9 shows disambiguation results averaged over the ten data sets.

From Table 4.9, we see that TEL performs the best, edging out the second place MDL+Default by a tiny margin, and followed by LA+Default, and SA+Default. I discuss these results below.

Table 4.9: PP-attachment disambiguation results.

Method	Coverage(%)	Accuracy(%)
Default	100	56.2
MDL + Default	100	82.2
SA + Default	100	76.7
LA + Default	100	80.7
LA.t + Default	100	78.1
TEL	100	82.4

MDL and SA

Experimental results show that the accuracy and coverage of MDL appear to be somewhat better than those of SA. Table 4.10 shows example generalization results for MDL (with classes with probability less than 0.05 discarded) and SA. Note that MDL tends to select a tree cut model closer to the root of the thesaurus. This is probably the key reason that MDL has a wider coverage than SA for the same degree of accuracy. One may be concerned that MDL may be 'over-generalizing' here, but as shown in Figure 4.8, this does not seem to degrade its disambiguation accuracy.

Another problem which must be dealt with concerning SA is how to increase the reliability of estimation. Since SA actually uses the ratio between two probability estimates, namely $\frac{\hat{P}(C|v,r)}{\hat{P}(C)}$, when one of the estimates is unreliably estimated, the ratio may be lead astray. For instance, the high estimated value shown in Table 4.10 for $\langle$drop,bead,pearl$\rangle$ at 'protect against' is rather odd, and arises because the estimate of $\hat{P}(C)$ is unreliable (very small). This problem apparently costs SA a non-negligible drop in the disambiguation accuracy.

MDL and LA

LA makes its disambiguation decision completely ignoring n_2. As (Resnik, 1993b) pointed out, if we hope to improve disambiguation performance with increasing training data, we need a richer model, such as those used in MDL and SA. I found that 8.8% of the quadruples in the entire test data were such that they shared the same (v, p, n_1) but had different n_2, and their pp-attachment sites went both ways in the same data, i.e., both to v and to n_1. Clearly, for these examples, the pp-attachment site cannot be reliably determined without knowing n_2. Table 4.11 shows some of these examples. (I have adopted the attachment sites given in the Penn Tree Bank, without correcting apparently wrong judgements.)

MDL and TEL

TEL seems to perform slightly better than MDL. We can, however, develop a more sophisticated MDL method which outperforms TEL, as may be seen in Chapter 7.

4.6 Summary

I have proposed a method for generalizing case slots. The method has the following merits: (1) it is theoretically sound; (2) it is computationally efficient; (3) it is robust against noise. One of the disadvantages of the method is that its performance depends on the structure of the particular thesaurus used. This, however, is a problem commonly shared by any generalization method which uses a thesaurus as prior knowledge.

The approach of applying MDL to estimate a tree cut model in an existing thesaurus is not limited to just the problem of generalizing values of a case slot. It is potentially useful in other natural language processing tasks, such as estimating n-gram models (cf., (Brown et al., 1992; Stolcke and Segal, 1994; Pereira and Singer, 1995; Rosenfeld, 1996; Ristad and Thomas, 1995; Saul and Pereira, 1997)) or semantic tagging (cf., (Cucchiarelli and Velardi, 1997)).

Table 4.10: Example generalization results for SA and MDL.

Input			
Verb	Preposition	Noun	Frequency
protect	against	accusation	1
protect	against	damage	1
protect	against	decline	1
protect	against	drop	1
protect	against	loss	1
protect	against	resistance	1
protect	against	squall	1
protect	against	vagary	1
Generalization result of MDL			
Verb	Preposition	Noun class	Probability
protect	against	⟨act, human_action, human_activity⟩	0.212
protect	against	⟨phenomenon⟩	0.170
protect	against	⟨psychological_feature⟩	0.099
protect	against	⟨event⟩	0.097
protect	against	⟨abstraction⟩	0.093
Generalization result of SA			
Verb	Preposition	Noun class	SA
protect	against	⟨caprice, impulse, vagary, whim⟩	1.528
protect	against	⟨phenomenon⟩	0.899
protect	against	⟨happening, occurrence, natural_event⟩	0.339
protect	against	⟨deterioration, worsening, decline, declination⟩	0.285
protect	against	⟨act, human_action, human_activity⟩	0.260
protect	against	⟨drop, bead, pearl⟩	0.202
protect	against	⟨drop⟩	0.202
protect	against	⟨descent, declivity, fall, decline, downslope⟩	0.188
protect	against	⟨resistor, resistance⟩	0.130
protect	against	⟨underground, resistance⟩	0.130
protect	against	⟨immunity, resistance⟩	0.124
protect	against	⟨resistance, opposition⟩	0.111
protect	against	⟨loss, deprivation⟩	0.105
protect	against	⟨loss⟩	0.096
protect	against	⟨cost, price, terms, damage⟩	0.052

Table 4.11: Some hard examples for LA.

Attached to v	Attached to n_1
acquire interest in year	acquire interest in firm
buy stock in trade	buy stock in index
ease restriction on export	ease restriction on type
forecast sale for year	forecast sale for venture
make payment on million	make payment on debt
meet standard for resistance	meet standard for car
reach agreement in august	reach agreement in principle
show interest in session	show interest in stock
win verdict in winter	win verdict in case

Chapter 5

Case Dependency Learning

The concept of the mutual independence of events is the most essential sprout in the development of probability theory.

- Andrei Kolmogorov

In this chapter, I describe one method for learning the case frame model, i.e., learning dependencies between case frame slots.

5.1 Dependency Forest Model

As described in Chapter 3, we can view the problem of learning dependencies between case slots for a given verb as that of learning a multi-dimensional discrete joint probability distribution referred to as a 'case frame model.' The number of parameters in a joint distribution will be exponential, however, if we allow interdependencies among all of the variables (even the slot-based case frame model has $O(2^n)$ parameters, where n is the number of random variables), and thus their accurate estimation may not be feasible in practice. It is often assumed implicitly in natural language processing that case slots (random variables) are *mutually independent*.

Although assuming that random variables are mutually independent would drastically reduce the number of parameters (e.g., under the independence assumption, the number of parameters in a slot-based model becomes $O(n)$). As illustrated in (1.2) in Chapter 1, this assumption is not necessarily valid in practice.

What seems to be true in practice is that some case slots are in fact dependent on one another, but that the overwhelming majority of them are mutually independent, due partly to the fact that usually only a few case slots are obligatory; the others are optional. (Optional case slots are not necessarily independent, but if two optional case slots are randomly selected, it is very likely that they are independent of one another.) Thus the target joint distribution is likely to be approximatable as the product of

lower order component distributions, and thus has in fact a reasonably small number of parameters. We are thus lead to the approach of approximating the target joint distribution by a simplified distribution based on corpus data.

In general, any n-dimensional discrete joint distribution can be written as

$$P(X_1, X_2, \cdots, X_n) = \prod_{i=1}^{n} P(X_{m_i}|X_{m_1}, \cdots, X_{m_{i-1}})$$

for a permutation $(m_1, m_2, \cdots, m_n)$ of $(1, 2, \cdots, n)$, letting $P(X_{m_1}|X_{m_0})$ denote $P(X_{m_1})$.

A plausible assumption regarding the dependencies between random variables is that each variable *directly* depends on at most one other variable. This is one of the simplest assumptions that can be made to relax the independence assumption. For example, if the joint distribution $P(X_1, X_2, X_3)$ over 3 random variables X_1, X_2, X_3 can be written (approximated) as follows, it (approximately) satisfies such an assumption:

$$P(X_1, X_2, X_3) = (\approx)P(X_1) \cdot P(X_2|X_1) \cdot P(X_3|X_2). \tag{5.1}$$

I call such a distribution a 'dependency forest model.'

A dependency forest model can be represented by a dependency forest (i.e., a set of dependency trees), whose nodes represent random variables (each labeled with a number of parameters), and whose directed links represent the dependencies that exist between these random variables. A dependency forest model is thus a restricted form of a Bayesian network (Pearl, 1988). Graph (5) in Figure 5.1 represents the dependency forest model defined in (5.1). Table 5.1 shows the parameters associated with each node in the graph, assuming that the dependency forest model is slot-based. When a distribution can be represented by a single dependency tree, I call it a 'dependency tree model.'

Table 5.1: Parameters labeled with each node.

Node	Parameters
X_1	$P(X_1 = 1)$, $P(X_1 = 0)$
X_2	$P(X_2 = 1\|X_1 = 1)$, $P(X_2 = 0\|X_1 = 1)$, $P(X_2 = 1\|X_1 = 0)$, $P(X_2 = 0\|X_1 = 0)$
X_3	$P(X_3 = 1\|X_2 = 1)$, $P(X_3 = 0\|X_2 = 1)$, $P(X_3 = 1\|X_2 = 0)$, $P(X_3 = 0\|X_2 = 0)$

It is not difficult to see that disregarding the actual values of the probability parameters, we will have 16 and only 16 dependency forest models (i.e., 16 dependency forests) as approximations of the joint distribution $P(X_1, X_2, X_3)$, Since some of them are equivalent with each other, they can be further reduced into 7 equivalent classes of dependency forest models. Figure 5.1 shows the 7 equivalent classes and their members. (It is easy to verify that the dependency tree models based on a 'labeled free

tree' are equivalent to one another (cf., Appendix A.5). Here a 'labeled free tree' refers to a tree in which each node is uniquely associated with a label and in which any node can be the root (Knuth, 1973).)

5.2 Algorithm

Now we turn to the problem of how to select the best dependency forest model from among all possible ones to approximate a target joint distribution based on the data. This problem has already been investigated in the area of machine learning and related fields. One classical method is Chow & Liu's algorithm for estimating a multi-dimensional discrete joint distribution as a dependency tree model, in a way which is both efficient and theoretically sound (Chow and Liu, 1968).[1] More recently, Suzuki has extended their algorithm, on the basis of the MDL principle, so that it estimates the target joint distribution as a dependency forest model (Suzuki, 1993), and Suzuki's is the algorithm I employ here.

Suzuki's algorithm first calculates the statistic θ between all node pairs. The statistic $\theta(X_i, X_j)$ between node X_i and X_j is defined as

$$\theta(X_i, X_j) = \hat{I}(X_i, X_j) - \frac{(k_i - 1) \cdot (k_j - 1)}{2} \cdot \log N,$$

where $\hat{I}(X_i, X_j)$ denotes the empirical mutual information between random variables X_i and X_j; k_i and k_j denote, respectively, the number of possible values assumed by X_i and X_j; and N the input data size. The empirical mutual information between random variables X_i and X_j is defined as

$$\begin{array}{l} \hat{I}(X_i, X_j) = \hat{H}(X_i) - \hat{H}(X_i|X_j) \\ \hat{H}(X_i) = -\sum_{x_i \in X_i} \hat{P}(x_i) \cdot \log \hat{P}(x_i) \\ \hat{H}(X_i|X_j) = -\sum_{x_i \in X_i} \sum_{x_j \in X_j} \hat{P}(x_i, x_j) \cdot \log \hat{P}(x_i|x_j), \end{array}$$

where $\hat{P}(.)$ denotes the maximum likelihood estimate of probability $P(.)$. Furthermore, $0 \times \log 0 = 0$ is assumed to be satisfied.

The algorithm then sorts the node pairs in descending order with respect to θ. It then puts a link between the node pair with the largest θ value, provided that this value is larger than zero. It repeats this process until no node pair is left unprocessed, provided that adding that link will not create a loop in the current dependency graph. Figure 5.2 shows the algorithm. Note that the dependency forest that is output by the algorithm may not be uniquely determined.

Concerning the above algorithm, the following proposition holds:

[1] In general, learning a Baysian network is an intractable task (Cooper and Herskovits, 1992).

Proposition 2 *The algorithm outputs a dependency forest model with the minimum description length.*

See Appendix A.6 for a proof of the proposition.

It is easy to see that the number of parameters in a dependency forest model is of the order $O(n \cdot k^2)$, where k is the maximum of all k_i, and n is the number of random variables. If we employ the 'quick sort algorithm' to perform line 4, average case time complexity of the algorithm will be only of the order $O(n^2 \cdot (k^2 + \log n))$, and worst case time complexity will be only of the order $O(n^2 \cdot (k^2 + n^2))$.

Let us now consider an example of how the algorithm works. Suppose that the input data is as given in Table 3.3 and there are 4 nodes (random variables) X_{arg1}, X_{arg2}, X_{from}, and X_{to}. Table 5.2 shows the statistic θ for all node pairs. The dependency forest shown in Figure 5.3 has been constructed on the basis of the values given in Table 5.2. The dependency forest indicates that there is dependency between the 'to' slot and the arg2 slot, and between the 'to' slot and the 'from' slot.

Table 5.2: The statistic θ for node pairs.

θ	X_{arg1}	X_{arg2}	X_{from}	X_{to}
X_{arg1}		-0.28	-0.16	-0.18
X_{arg2}			$\underline{0.11}$	$\underline{0.57}$
X_{from}				$\underline{0.28}$
X_{to}				

As previously noted, the algorithm is based on the MDL principle. In the current problem, a simple model means a model with fewer dependencies, and thus MDL provides a theoretically sound way to learn only those dependencies that are statistically significant in the given data. As mentioned in Chapter 2, an especially interesting feature of MDL is that it incorporates the input data size in its model selection criterion. This is reflected, in this case, in the derivation of the threshold θ. Note that when we do not have enough data (i.e., N is too small), the thresholds will be large and few nodes will be linked, resulting in a simpler model in which most of the random variables are judged to be mutually independent. This is reasonable since with a small data size most random variables cannot be determined to be dependent with any significance.

Since the number of dependency forest models for a fixed number of random variables n is of order $O(2^{n-1} \cdot n^{n-2})$ (the number of dependency tree models is of order $\Theta(n^{n-2})$ (Knuth, 1973)), it would be impossible to calculate description length straightforwardly for all of them. Suzuki's algorithm effectively utilizes the *tree structures* of the models and efficiently calculates description lengths by doing it locally (as does Chow & Liu's algorithm).

5.3 Experimental Results

I have experimentally tested the performance of the proposed method of learning dependencies between case slots. Most specifically, I have tested to see how effective the dependencies acquired by the proposed method are when used in disambiguation experiments. In this section, I describe the procedures and the results of those experiments.

5.3.1 Experiment 1: slot-based model

In the first experiment, I tried to learn slot-based dependencies. As training data, I used the entire bracketed data of the Wall Street Journal corpus (Penn Tree Bank). I extracted case frame data from the corpus using heuristic rules. There were 354 verbs for which more than 50 case frame instances were extracted from the corpus. Table 5.3 shows the most frequent verbs and the corresponding numbers of case frames. In the experiment, I only considered the 12 most frequently occurring case slots (shown in Table 5.4) and ignored others.

Table 5.3: Verbs appearing most frequently.

Verb	Number of case frames
be	17713
say	9840
have	4030
make	1770
take	1245
expect	1201
sell	1147
rise	1125
get	1070
go	1042
do	982
buy	965
fall	862
add	740
come	733
include	707
give	703
pay	700
see	680
report	674

Table 5.4: Case slots considered.

arg1	arg2	on	in	for	at
by	from	to	as	with	against

Example case frame patterns

I acquired slot-based case frame patterns for the 354 verbs. There were on average $484/354 = 1.4$ dependency links acquired for each of these 354 verbs. As an example, Figure 5.4 shows the case frame patterns (dependency forest model) obtained for the verb 'buy.' There are four dependencies in this model; one indicates that, for example, the arg2 slot is dependent on the arg1 slot.

I found that there were some verbs whose arg2 slot is dependent on a preposition (hereafter, p for short) slot. Table 5.5 shows the 40 verbs having the largest values of $P(X_{\mathrm{arg2}} = 1, X_p = 1)$, sorted in descending order of these values. The dependencies found by the method seem to agree with human intuition.

Furthermore, I found that there were some verbs having preposition slots that depend on each other (I refer to these as $p1$ and $p2$ for short). Table 5.6 shows the 40 verbs having the largest values of $P(X_{p1} = 1, X_{p2} = 1)$, sorted in descending order. Again, the dependencies found by the method seem to agree with human intuition.

Perplexity reduction

I also evaluated the acquired case frame patterns (slot-based models) for all of the 354 verbs in terms of reduction of the 'test data perplexity.'[2]

I conducted the evaluation through a ten-fold cross validation. That is, to acquire case frame patterns for the verb, I used nine tenths of the case frames for each verb as training data, saving what remained for use as test data, and then calculated the test data perplexity. I repeated this process ten times and calculated average perplexity. I also calculated average perplexity for 'independent models' which were acquired based on the assumption that each case slot is independent.

Experimental results indicate that for some verbs the use of the dependency forest model results in less perplexity than does use of the independent model. For 30 of the 354 (8%) verbs, perplexity reduction exceeded 10%, while average perplexity reduction overall was 1%. Table 5.7 shows the 10 verbs having the largest perplexity reductions. Table 5.8 shows perplexity reductions for 10 randomly selected verbs. There were a

[2]The test data perplexity is a measure of testing how well an estimated probability model predicts future data, and is defined as $2^{H(P_T,P_M)}$, $H(P_T,P_M) = -\sum_x P_T(x) \cdot \log P_M(x)$, where $P_M(x)$ denotes the estimated model, $P_T(x)$ the empirical distribution of the test data (cf., (Bahl, Jelinek, and Mercer, 1983)). It is roughly the case that the smaller perplexity a model has, the closer to the true model it is.

Table 5.5: Verbs and their dependent case slots.

Verb	Dependent slots	Example
base	arg2 on	base pay on education
advance	arg2 to	advance 4 to 40
gain	arg2 to	gain 10 to 100
compare	arg2 with	compare profit with estimate
invest	arg2 in	invest share in fund
acquire	arg2 for	acquire share for billion
estimate	arg2 at	estimate price at million
convert	arg2 to	convert share to cash
add	arg2 to	add 1 to 3
engage	arg2 in	enage group in talk
file	arg2 against	file suit against company
aim	arg2 at	aim it at transaction
sell	arg2 to	sell facility to firm
lose	arg2 to	lose million to 10%
pay	arg2 for	pay million for service
leave	arg2 with	leave himself with share
charge	arg2 with	charge them with fraud
provide	arg2 for	provide engine for plane
withdraw	arg2 from	withdraw application from office
prepare	arg2 for	prepare case for trial
succeed	arg2 as	succeed Taylor as chairman
discover	arg2 in	discover mile in ocean
move	arg2 to	move employee to New York
concentrate	arg2 on	concentrate business on steel
negotiate	arg2 with	negotiate rate with advertiser
open	arg2 to	open market to investor
protect	arg2 against	protect investor against loss
keep	arg2 on	keep eye on indicator
describe	arg2 in	describe item in inch
see	arg2 as	see shopping as symptom
boost	arg2 by	boost value by 2%
pay	arg2 to	pay commission to agent
contribute	arg2 to	contribute million to leader
bid	arg2 for	bid million for right
threaten	arg2 against	threaten sanction against lawyer
file	arg2 for	file lawsuit for dismissal
know	arg2 as	know him as father
sell	arg2 at	sell stock at time
settle	arg2 at	settle session at 99
see	arg2 in	see growth in quarter

small number of verbs showing perplexity increases with the worst case being 5%. It seems safe to say that the dependency forest model is more suitable for representing the 'true' model of case frames than the independent model, at least for 8% of the 354 verbs.

5.3.2 Experiment 2: slot-based disambiguation

To evaluate the effectiveness of the use of dependency knowledge in natural language processing, I conducted a pp-attachment disambiguation experiment. Such disambiguation would be, for example, to determine which word, 'fly' or 'jet,' the phrase 'from Tokyo' should be attached to in the sentence "She will fly a jet from Tokyo." A straightforward way of disambiguation would be to compare the following likelihood values, based on slot-based models,

$$P_{\text{fly}}(X_{\text{arg2}} = 1, X_{\text{from}} = 1) \cdot P_{\text{jet}}(X_{\text{from}} = 0)$$

and

$$P_{\text{fly}}(X_{\text{arg2}} = 1, X_{\text{from}} = 0) \cdot P_{\text{jet}}(X_{\text{from}} = 1),$$

assuming that there are only two case slots: arg2 and 'from' for the verb 'fly,' and there is one case slot: 'from' for the noun 'jet.' In fact, we need only compare

$$P_{\text{fly}}(X_{\text{from}} = 1|X_{\text{arg2}} = 1) \cdot (1 - P_{\text{jet}}(X_{\text{from}} = 1))$$

and

$$(1 - P_{\text{fly}}(X_{\text{from}} = 1|X_{\text{arg2}} = 1)) \cdot P_{\text{jet}}(X_{\text{from}} = 1),$$

or equivalently,

$$P_{\text{fly}}(X_{\text{from}} = 1|X_{\text{arg2}} = 1)$$

and

$$P_{\text{jet}}(X_{\text{from}} = 1).$$

Obviously, if we assume that the case slots are independent, then we need only compare $P_{\text{fly}}(X_{\text{from}} = 1)$ and $P_{\text{jet}}(X_{\text{from}} = 1)$. This is equivalent to the method proposed by (Hindle and Rooth, 1991). Their method actually compares the two probabilities by means of hypothesis testing.

It is here that we first employ the proposed dependency learning method to judge if slots X_{arg2} and X_{from} with respect to verb 'fly' are mutually dependent; if they are dependent, we make a disambiguation decision based on the t-score between $P_{\text{fly}}(X_{\text{from}} = 1|X_{\text{from}} = 1)$ and $P_{\text{jet}}(X_{\text{from}} = 1)$; otherwise, we consider the two slots independent and make a decision based on the t-score between $P_{\text{fly}}(X_{\text{from}} = 1)$ and $P_{\text{jet}}(X_{\text{from}} = 1)$. I refer to this method as 'DepenLA.'

In the experiment, I first randomly selected the files under one directory for a portion of the WSJ corpus, a portion containing roughly one 26th of the entire bracketed

corpus data, and extracted (v, n_1, p, n_2) quadruples (e.g., (fly, jet, from, Tokyo)) as test data. I then extracted case frames from the remaining bracketed corpus data as I did in Experiment 1 and used them as training data. I repeated this process ten times and obtained ten data sets consisting of different training data and test data. In each training data set, there were roughly $128,000$ case frames on average for verbs and roughly $59,000$ case frames for nouns. On average, there were 820 quadruples in each test data set.

I used these ten data sets to conduct disambiguation through cross validation. I used the training data to acquire dependency forest models, which I then used to perform disambiguation on the test data on the basis of DepenLA. I also tested the method of LA. I set the threshold for the t-score to 1.28. For both LA and DepenLA, there were still some quadruples remaining whose attachment sites could not be determined. In such cases, I made a default decision, i.e., forcibly attached (p, n_2) to v, because I empirically found that, at least for our data set for what remained after applying LA and DepenLA, it is more likely for (p, n_2) to go with v. Tab. 5.9 summarizes the results, which are evaluated in terms of disambiguation accuracy, averaged over the ten trials.

I found that as a whole DepenLA+Default only slightly improves LA+Default. I further found, however, that for about 11% of the data in which the dependencies are strong (i.e., $P(X_p = 1|X_{\text{arg2}} = 1) > 0.2$ or $P(X_p = 1|X_{\text{arg2}} = 1) < 0.002$), DepenLA+Default significantly improves LA+Default. That is to say that when significant dependencies between case slots are found, the disambiguation results can be improved by using dependency knowledge. These results to some extent agree with the perplexity reduction results obtained in Experiment 1.

5.3.3 Experiment 3: class-based model

I also used the 354 verbs in Experiment 1 to acquire case frame patterns as class-based dependency forest models. Again, I considered only the 12 slots listed in Table 5.4. I generalized the values of the case slots within these case frames using the method proposed in Chapter 4 to obtain class-based case frame data like those presented in Table 3.3.[3] I used these data as input to the learning algorithm.

On average, there was only a $64/354 = 0.2$ dependency link found in the patterns for a verb. That is, very few case slots were determined to be dependent in the case frame patterns. This is because the number of parameters in a class based model was larger than the size of the data we had available.

The experimental results indicate that it is often valid in practice to assume that class-based case slots (and also word-based case slots) are mutually independent, when the data size available is at the level of what is provided by Penn Tree Bank. For this

[3]Since a node in WordNet represents a word sense and not a word, a word can belong to several different classes (nodes) in an output tree cut model. I have heuristically replaced a word n with the word class C such that $\max_{C \ni n}(P(C|v, r))$ is satisfied.

reason, I did not conduct disambiguation experiments using the class-based dependency forest models.

I believe that the proposed method provides a theoretically sound and effective tool for detecting whether there exists a statistically significant dependency between case slots in given data; this decision has up to now been based simply on human intuition.

5.3.4 Experiment 4: simulation

In order to test how large a data size is required to estimate a dependency forest model, I conducted the following experiment. I defined an artificial model in the form of a dependency forest model and generated data on the basis of its distribution. I then used the obtained data to estimate a model, and evaluated the estimated model by measuring the KL divergence between the estimated model and the true model. I also checked the number of dependency links in the obtained model. I repeatedly generated data and observed the 'learning curve,' namely the relationship between the data size used in estimation and the number of links in the estimated model, and the relationship between the data size and the KL divergence separating the estimated and the true model. I defined two other artificial models and conducted the same experiments. Figures 5.5 and 5.6 show the results of these experiments for the three artificial models averaged over 10 trials. The number of parameters in Model 1, Model 2, and Model 3 are 18, 30, and 44 respectively, and the number of links in them 1, 3, and 5. Note that the KL divergences between the estimated models and the true models converge to 0, as expected. Also note that the numbers of links in the estimated models converge to the correct value (1, 3, and 5) in each of the three examples.

These simulation results verify the consistency property of MDL (i.e., the numbers of parameters in the selected models converge in probability to that of the true model as the data size increases), which is crucial for the goal of learning dependencies. Thus we can be confident that the dependencies between case slots can be accurately learned when there are enough data, as long as the 'true' model exists as a dependency forest model.

We also see that to estimate a model accurately the data size required is as large as 5 to 10 times the number of parameters. For example, for the KL divergence to go to below 0.1, we need more than 200 examples, which is roughly 5 to 10 times the number of parameters.

Note that in Experiment 3, I considered 12 slots, and for each slot there were roughly 10 classes as its values; thus a class-based model tended to have about 120 parameters. The corpus data available to us was insufficient for accurate learning of the dependencies between case slots for most verbs (cf., Table 5.3).

5.4 Summary

I conclude this chapter with the following remarks.

1. The primary contribution of the research reported in this chapter is the proposed method of learning dependencies between case slots, which is theoretically sound and efficient.

2. For slot-based models, some case slots are found to be dependent. Experimental results demonstrate that by using the knowledge of dependency, when dependency does exist, we can significantly improve pp-attachment disambiguation results.

3. For class-based models, most case slots are judged independent with the data size currently available in the Penn Tree Bank. This empirical finding indicates that it is often valid to assume that case slots in a class-based model are mutually independent.

The method of using a dependency forest model is not limited to just the problem of learning dependencies between case slots. It is potentially useful in other natural language processing tasks, such as word sense disambiguation (cf., ((Bruce and Wiebe, 1994))).

$$\begin{tikzcd} & X_1 & \\ X_2 & & X_3 \end{tikzcd}$$

$$\begin{aligned} & P(X_1, X_2, X_3) \\ &= P(X_1)P(X_2)P(X_3) \end{aligned}$$

(1)

$$\begin{tikzcd} & X_1 & \\ X_2 \arrow[rr] & & X_3 \end{tikzcd}$$

$$\begin{aligned} & P(X_1, X_2, X_3) \\ &= P(X_1)P(X_2)P(X_3|X_2) \\ &= P(X_1)P(X_3)P(X_2|X_3) \end{aligned}$$

(2)

$$\begin{tikzcd} & X_1 \arrow[dl] & \\ X_2 & & X_3 \end{tikzcd}$$

$$\begin{aligned} & P(X_1, X_2, X_3) \\ &= P(X_1)P(X_2|X_1)P(X_3) \\ &= P(X_2)P(X_1|X_2)P(X_3) \end{aligned}$$

(3)

$$\begin{tikzcd} & X_1 \arrow[dr] & \\ X_2 & & X_3 \end{tikzcd}$$

$$\begin{aligned} & P(X_1, X_2, X_3) \\ &= P(X_1)P(X_3|X_1)P(X_2) \\ &= P(X_3)P(X_1|X_3)P(X_2) \end{aligned}$$

(4)

$$\begin{tikzcd} & X_1 \arrow[dl] & \\ X_2 \arrow[rr] & & X_3 \end{tikzcd}$$

$$\begin{aligned} & P(X_1, X_2, X_3) \\ &= P(X_1)P(X_2|X_1)P(X_3|X_2) \\ &= P(X_2)P(X_1|X_2)P(X_3|X_2) \\ &= P(X_3)P(X_2|X_3)P(X_1|X_2) \end{aligned}$$

(5)

$$\begin{tikzcd} & X_1 \arrow[dr] & \\ X_2 \arrow[ur] & & X_3 \end{tikzcd}$$

$$\begin{aligned} & P(X_1, X_2, X_3) \\ &= P(X_2)P(X_1|X_2)P(X_3|X_1) \\ &= P(X_1)P(X_3|X_1)P(X_2|X_1) \\ &= P(X_3)P(X_1|X_3)P(X_2|X_1) \end{aligned}$$

(6)

$$\begin{tikzcd} & X_1 \arrow[dr] & \\ X_2 & & X_3 \arrow[ll] \end{tikzcd}$$

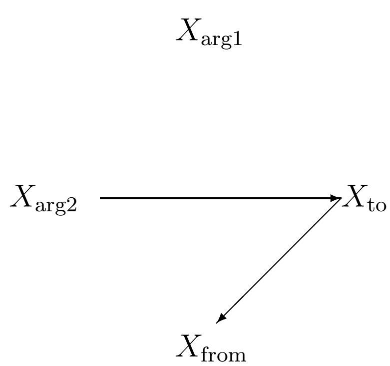
$$\begin{aligned} & P(X_1, X_2, X_3) \\ &= P(X_1)P(X_3|X_1)P(X_2|X_3) \\ &= P(X_3)P(X_1|X_3)P(X_2|X_3) \\ &= P(X_2)P(X_3|X_2)P(X_1|X_3) \end{aligned}$$

(7)

Figure 5.1: Example dependency forests.

Algorithm:
1. Let $T := \emptyset$;
2. Let $V = \{\{X_i\}, i = 1, 2, \cdots, n\}$;
3. Calculate $\theta(X_i, X_j)$ for all node pairs (X_i, X_j) ;
4. Sort the node pairs in descending order of θ, and store them into queue Q;
5. **while**
6. $\quad \max_{(X_i,X_j)\in Q} \theta(X_i, X_j) > 0$
7. **do**
8. $\quad$ Remove $\arg\max_{(X_i,X_j)\in Q} \theta(X_i, X_j)$ from Q;
9. $\quad$ **if**
10. $\qquad X_i$ and X_j belong to different sets W_1,W_2 in V
11. $\quad$ **then**
12. $\qquad$ Replace W_1 and W_2 in V with $W_1 \cup W_2$, and add edge (X_i, X_j) to T;
13. Output T as the set of edges of the dependency forest.

Figure 5.2: The learning algorithm.



$$
\begin{aligned}
&P(X_{\text{arg1}}, X_{\text{arg2}}, X_{from}, X_{\text{to}}) \\
&= P(X_{\text{arg1}})P(X_{\text{arg2}})P(X_{\text{to}}|X_{\text{arg2}})P(X_{\text{from}}|X_{\text{to}})
\end{aligned}
$$

Figure 5.3: A dependency forest as case frame patterns.

```
buy:
[arg1]: [P(arg1=0)=0.004 P(arg1=1)=0.996]
[arg2]: [P(arg2=0|arg1=0)=0.100,P(arg2=1|arg1=0)=0.900,
         P(arg2=0|arg1=1)=0.136,P(arg2=1|arg1=1)=0.864]
[for]: [P(for=0|arg1=0)=0.300,P(for=1|arg1=0)=0.700,
        P(for=0|arg1=1)=0.885,P(for=1|arg1=1)=0.115]
[at]: [P(at=0|for=0)=0.911,P(at=1|for=0)=0.089,
         P(at=0|for=1)=0.979,P(at=1|for=1)=0.021]
[in]: [P(in=0|at=0)=0.927,P(in=0|at=0)=0.073,
         P(in=0|at=1)=0.994,P(in=1|at=1)=0.006]
[on]: [P(on=0)=0.975,P(on=1)=0.025]
[from]: [P(from=0)=0.937,P(from=1)=0.063]
[to]: [P(to=0)=0.997,P(on=1)=0.003]
[by]: [P(by=0)=0.995,P(by=1)=0.005]
[with]: [P(with=0)=0.993,P(with=1)=0.007]
[as]: [P(as=0)=0.991,P(as=1)=0.009]
[against]: [P(against=0)=0.999,P(against=1)=0.001]
```

Figure 5.4: Case frame patterns (dependency forest model) for 'buy.'

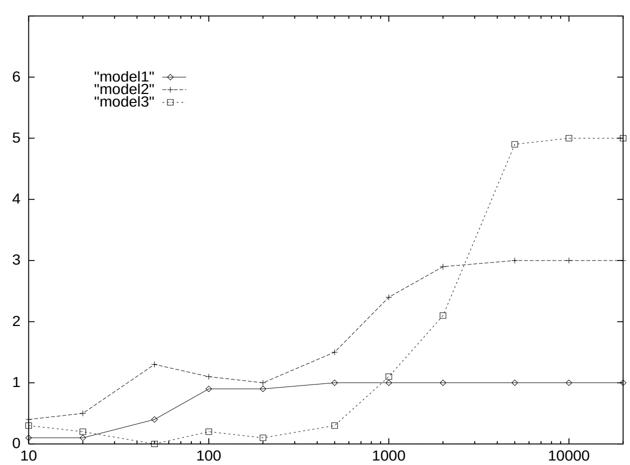


Figure 5.5: Number of links versus data size.

Table 5.6: Verbs and their dependent case slots.

Head	Dependent slots	Example
range	from to	range from 100 to 200
climb	from to	climb from million to million
rise	from to	rise from billion to billion
shift	from to	shift from stock to bond
soar	from to	soar from 10% to 20%
plunge	from to	plunge from 20% to 2%
fall	from to	fall from million to million
surge	from to	surge from 100 to 200
increase	from to	increase from million to million
jump	from to	jump from yen to yen
yield	from to	yield from 1% to 5%
climb	from in	climb from million in period
apply	to for	apply to commission for permission
grow	from to	grow from million to million
draw	from in	draw from thrift in bonus
boost	from to	boost from 1% to 2%
convert	from to	convert from form to form
raise	from to	raise from 5% to 10%
retire	on as	retire on 2 as officer
move	from to	move from New York to Atlanta
cut	from to	cut from 700 to 200
sell	to for	sell to bakery for amount
open	for at	open for trading at yen
lower	from to	lower from 10% to 2%
rise	to in	rise to 5% in month
trade	for in	trade for use in amount
supply	with by	supply with meter by 1990
elect	to in	elect to congress in 1978
point	to as	point to contract as example
drive	to in	drive to clinic in car
vote	on at	vote on proposal at meeting
acquire	from for	acquire from corp. for million
end	at on	end at 95 on Friday
apply	to in	apply to congress in 1980
gain	to on	gain to 3 on share
die	on at	die on Sunday at age
bid	on with	bid on project with Mitsubishi
file	with in	file with ministry in week
slow	from to	slow from pound to pound
improve	from to	improve from 10% to 50%

Table 5.7: Verbs with significant perplexity reduction.

Verb	Independent	Dependency forest (reduction in percentage)
base	5.6	3.6(36%)
lead	7.3	4.9(33%)
file	16.4	11.7(29%)
result	3.9	2.8(29%)
stem	4.1	3.0(28%)
range	5.1	3.7(28%)
yield	5.4	3.9(27%)
benefit	5.6	4.2(26%)
rate	3.5	2.6(26%)
negotiate	7.2	5.6(23%)

Table 5.8: Randomly selected verbs and their perplexities.

Verb	Independent	Dependency forest (reduction in percentage)
add	4.2	3.7(9%)
buy	1.3	1.3(0%)
find	3.2	3.2(0%)
open	13.7	12.3(10%)
protect	4.5	4.7(−4%)
provide	4.5	4.3(4%)
represent	1.5	1.5(0%)
send	3.8	3.9(−2%)
succeed	3.7	3.6(4%)
tell	1.7	1.7(0%)

Table 5.9: PP-attachment disambiguation results.

Method	Accuracy(%)
Default	56.2
LA+Default	78.1
DepenLA+Default	78.4
LA+Default(11% of data)	93.8
DepenLA+Default(11% of data)	97.5

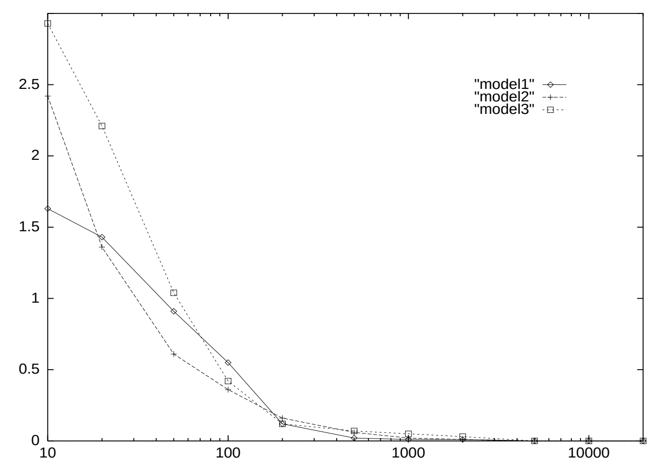


Figure 5.6: KL divergence versus data size.

Chapter 6

Word Clustering

We may add that objects can be classified, and can become similar or dissimilar, only in this way - by being related to needs and interests.

- Karl Popper

In this chapter, I describe one method for learning the hard co-occurrence model, i.e., clustering of words on the basis of co-occurrence data. This method is a natural extension of that proposed by Brown et al (cf., Chapter 2), and it overcomes the drawbacks of their method while retaining its merits.

6.1 Parameter Estimation

As described in Chapter 3, we can view the problem of clustering words (constructing a thesaurus) on the basis of co-occurrence data as that of estimating a hard co-occurrence model.

The fixing of partitions determines a discrete hard co-occurrence model and the number of parameters. We can estimate the values of the parameters on the basis of co-occurrence data by employing Maximum Likelihood Estimation (MLE). For given co-occurrence data

$$\mathcal{S} = \{(n_1, v_1), (n_2, v_2), \cdots, (n_m, v_m)\},$$

where n_i $(i = 1, \cdots, m)$ denotes a noun, and v_i $(i = 1, \cdots, m)$ a verb. The maximum likelihood estimates of the parameters are defined as the values that maximize the following likelihood function with respect to the data:

$$\prod_{i=1}^{m} P(n_i, v_i) = \prod_{i=1}^{m} (P(n_i|C_{n_i}) \cdot P(v_i|C_{v_i}) \cdot P(C_{n_i}, C_{v_i})).$$

It is easy to verify that we can estimate the parameters as

$$\hat{P}(C_n, C_v) = \frac{f(C_n, C_v)}{m}$$

$$\hat{P}(n|C_n) = \frac{f(n)}{f(C_n)}$$

$$\hat{P}(v|C_v) = \frac{f(v)}{f(C_v)},$$

so as to maximize the likelihood function, under the conditions that the sum of the joint probabilities over noun classes and verb classes equals one, and that the sum of the conditional probabilities over words in each class equals one. Here, m denotes the entire data size, $f(C_n, C_v)$ the frequency of word pairs in class pair (C_n, C_v), $f(n)$ the frequency of noun n, $f(v)$ that of v, $f(C_n)$ the frequency of words in class C_n, and $f(C_v)$ that in C_v.

6.2 MDL as Strategy

I again adopt the MDL principle as a strategy for statistical estimation.

Data description length may be calculated as

$$L(\mathcal{S}|M) = -\sum_{(n,v)\in\mathcal{S}} \log \hat{P}(n, v).$$

Model description length may be calculated, here, as

$$L(M) = \frac{k}{2} \cdot \log m,$$

where k denotes the number of *free* parameters in the model, and m the data size. We in fact implicitly assume here that the description length for encoding the discrete model is equal for all models and view only the description length for encoding the parameters as the model description length. Note that there are alternative ways of calculating the model description length. Here, for efficiency in clustering, I use the simplest formulation.

If computation time were of no concern, we could in principle calculate the total description length for each model and select the optimal model in terms of MDL. However, since the number of hard co-occurrence models is of order $O(N^N \cdot V^V)$ (cf., Chapter 4), where N and V denote the sizes of the set of nouns and the set of verbs respectively, it would be infeasible to do so. We therefore need to devise an efficient algorithm that will heuristically perform this task.

6.3 Algorithm

The algorithm that we have devised, denoted here as '2D-Clustering,' iteratively selects a suboptimal MDL model from among a class of hard co-occurrence models. These models include the current model and those which can be obtained from the current model by merging a noun (or verb) class pair. The minimum description length criterion can be reformalized in terms of (empirical) mutual information. The algorithm can be formulated as one which calculates, in each iteration, the reduction of mutual information which would result from merging any noun (or verb) class pair. It would perform the merge having the least mutual information reduction, *provided that the least mutual information reduction is below a threshold, which will vary depending on the data size and the number of classes in the current situation.*

2D-Clustering($\mathcal{S}$)
$\mathcal{S}$ is input co-occurrence data. b_n and b_v are positive integers.

1. Initialize the set of noun classes Π_n and the set of verb classes Π_v as:

$$\Pi_n = \{\{n\}|n \in \mathcal{N}\}$$

$$\Pi_v = \{\{v\}|v \in \mathcal{V}\}$$

   $\mathcal{N}$ and $\mathcal{V}$ denote the set of nouns and the set of verbs, respectively.

2. Repeat the following procedure:
   (a) execute Merge($\mathcal{S}, \Pi_n, \Pi_v, b_n$) to update Π_n,
   (b) execute Merge($\mathcal{S}, \Pi_v, \Pi_n, b_v$) to update Π_v,
   (c) if Π_n and Π_v are unchanged, go to Step 3.

3. Construct and output a thesaurus of nouns based on the history of Π_n, and one for verbs based on the history of Π_v.

For the sake of simplicity, let us next consider only the procedure for Merge as it is applied to the set of noun classes while the set of verb classes is fixed.

Merge($\mathcal{S}, T_n, T_v, b_n$)

1. For each class pair in T_n, calculate the reduction in mutual information which would result from merging them. (Details of such a calculation are given below.) Discard those class pairs whose mutual information reduction is not less than the threshold of

$$\frac{(k_B - k_A) \cdot \log m}{2 \cdot m}, \tag{6.1}$$

where m denotes total data size, k_B the number of free parameters in the model before the merge, and k_A the number of free parameters in the model after the merge. Sort the remaining class pairs in ascending order with respect to mutual information reduction.

2. Merge the first b_n class pairs in the sorted list.

3. Output current T_n.

For improved efficiency, the algorithm performs a maximum of b_n merges at step 2, which will result in the output of an at most b_n-ary tree. Note that, strictly speaking, once we perform one merge, the model will change and there will no longer be any guarantee that the remaining merges continue to be justifiable from the viewpoint of MDL.

Next, let us consider why the criterion formalized in terms of description length can be reformalized in terms of mutual information. Let M_B refer to the pre-merge model, M_A to the post-merge model. According to MDL, M_A should be that model which has the least increase in data description length

$$\delta L_{dat} = L(\mathcal{S}|M_A) - L(\mathcal{S}|M_B) > 0$$

and that at the same time satisfies

$$\delta L_{dat} < \frac{(k_B - k_A) \cdot \log m}{2},$$

since the decrease in model description length equals

$$L(M_B) - L(M_A) = \frac{(k_B - k_A) \cdot \log m}{2} > 0,$$

and the decrease in model description length is common to each merge.

In addition, suppose that M_A is obtained by merging two noun classes C_i and C_j in M_B to a noun class C_{ij}. We in fact need only calculate the difference in description lengths with respect to these classes, i.e.,

$$\begin{aligned}\delta L_{dat} &= -\textstyle\sum_{C_v \in \Pi_v} \sum_{n \in C_{ij}, v \in C_v} \log \hat{P}(n,v) + \sum_{C_v \in \Pi_v} \sum_{n \in C_i, v \in C_v} \log \hat{P}(n,v) \\ &\quad + \textstyle\sum_{C_v \in \Pi_v} \sum_{n \in C_j, v \in C_v} \log \hat{P}(n,v).\end{aligned}$$

Since

$$P(n,v) = \frac{P(n)}{P(C_n)} \cdot \frac{P(v)}{P(C_v)} \cdot P(C_n, C_v) = \frac{P(C_n, C_v)}{P(C_n)P(C_v)} \cdot P(n) \cdot P(v)$$

holds, we also have

$$\hat{P}(n) = \frac{f(n)}{m},$$

$$\hat{P}(v) = \frac{f(v)}{m},$$

$$\hat{P}(C_n) = \frac{f(C_n)}{m},$$

and

$$\hat{P}(C_v) = \frac{f(C_v)}{m}.$$

Hence,

$$\begin{aligned} \delta L_{dat} &= -\sum_{C_v \in \Pi_v} f(C_{ij}, C_v) \cdot \log \frac{\hat{P}(C_{ij}, C_v)}{\hat{P}(C_{ij})\hat{P}(C_v)} + \sum_{C_v \in \Pi_v} f(C_i, C_v) \cdot \log \frac{\hat{P}(C_i, C_v)}{\hat{P}(C_i)\hat{P}(C_v)} \\ &+ \sum_{C_v \in \Pi_v} f(C_j, C_v) \cdot \log \frac{\hat{P}(C_j, C_v)}{\hat{P}(C_j)\hat{P}(C_v)}. \end{aligned} \tag{6.2}$$

The quantity δL_{dat} is equivalent to the data size times the empirical mutual information reduction. We can, therefore, say that in the current context a clustering with the least data description length increase is equivalent to that with the least mutual information decrease.

Note further that in (6.2), since $\hat{P}(C_v)$ is unchanged before and after the merge, it can be canceled out. Replacing the probabilities with their maximum likelihood estimates, we obtain

$$\begin{aligned} \frac{1}{m} \cdot \delta L_{dat} &= \frac{1}{m} \cdot \Big(-\sum_{C_v \in \Pi_v} (f(C_i, C_v) + f(C_j, C_v)) \cdot \log \frac{f(C_i, C_v) + f(C_j, C_v)}{f(C_i) + f(C_j)} \\ &+ \sum_{C_v \in \Pi_v} f(C_i, C_v) \cdot \log \frac{f(C_i, C_v)}{f(C_i)} + \sum_{C_v \in \Pi_v} f(C_j, C_v) \cdot \log \frac{f(C_j, C_v)}{f(C_j)} \Big). \end{aligned}$$

We need calculate only this quantity for each possible merge at Step 1 in Merge.

In an implementation of the algorithm, we first load the co-occurrence data into a matrix, with nouns corresponding to rows, verbs to columns. When merging a noun class in row i and that in row j ($i < j$), for each C_v, we add $f(C_i, C_v)$ and $f(C_j, C_v)$, obtaining $f(C_{ij}, C_v)$; then write $f(C_{ij}, C_v)$ on row i; move $f(C_{last}, C_v)$ to row j. This reduces the matrix by one row.

With the above implementation, the worst case time complexity of the algorithm turns out to be $O(N^3 \cdot V + V^3 \cdot N)$, where N denotes the size of the set of nouns, and V that of verbs. If we can merge b_n and b_v classes at each step, the algorithm will become slightly more efficient, with a time complexity of $O(\frac{N^3}{b_n} \cdot V + \frac{V^3}{b_v} \cdot N)$.

The method proposed in this chapter is an extension of that proposed by Brown et al. Their method iteratively merges the word class pair having the least reduction in mutual information until the number of word classes created equals a certain designated number. This method is based on MLE, but it only employs MLE *locally*.

In general, MLE is not able to select the best model from a class of models having different numbers of parameters because MLE will always suggest selecting the model having the largest number of parameters, which would have a better fit to the given data. In Brown et al's case, MLE is used to iteratively select the model with the

maximum likelihood from a class of *models that have the same number of parameters*. Such a model class is repeatedly obtained by merging any word class pair in the current situation. The number of word classes within the models in the final model class, therefore, has to be designated in advance. There is, however, no guarantee at all the designated number will be optimal.

The method proposed here resolves this problem by employing MDL. This is reflected in use of the threshold (6.1) in clustering, which will result in automatic selection of the optimal number of word classes to be created.

6.4 Experimental Results

6.4.1 Experiment 1: qualitative evaluation

In this experiment, I used heuristic rules to extract verbs and their arg2 slot values (direct objects) from the *tagged* texts of the WSJ corpus (ACL/DCI CD-ROM1) which consists of 126,084 sentences.

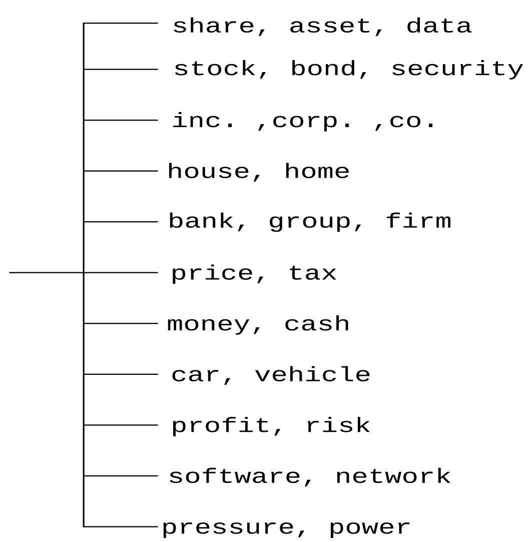


Figure 6.1: A part of a constructed thesaurus.

I then constructed a number of thesauruses based on these data, using the method proposed in this chapter. Figure 6.1 shows a part of a thesaurus for 100 randomly selected nouns, that serve as direct objects of 20 randomly selected verbs. The thesaurus seems to agree with human intuition to some degree. The words 'stock,' 'security,' and 'bond' are classified together, for example, despite the fact that their absolute frequencies are quite different (272, 59, and 79, respectively). The results seem to demonstrate

one desirable feature of the proposed method: it classifies words solely on the basis of the similarities in co-occurrence data and is not affected by the absolute frequencies of the words.

6.4.2 Experiment 2: compound noun disambiguation

I tested the effectiveness of the clustering method by using the acquired word classes in compound noun disambiguation. This would determine, for example, the word 'base' or 'system' to which 'data' should be attached in the compound noun triple (data, base, system).

To conduct compound noun disambiguation, we can use here the probabilities

$$\hat{P}(\text{data}|\text{base}), \tag{6.3}$$

$$\hat{P}(\text{data}|\text{system}). \tag{6.4}$$

If the former is larger, we attach 'data' to 'base;' if the latter is larger we attach it to 'system;' otherwise, we make no decision.

I first randomly selected 1000 nouns from the corpus, and extracted from the corpus compound noun doubles (e.g., (data, base)) containing the nouns as training data and compound noun triples containing the nouns as test data. There were 8604 training data and 299 test data. I also labeled the test data with disambiguation 'answers.'

I conducted clustering on the nouns in the left position in the training data, and also on the nouns in the right position, by using, respectively, both the method proposed in this chapter, denoted as '2D-Clustering,' and Brown et al's, denoted as 'Brown.' I actually implemented an extended version of their method, which separately conducts clustering for nouns on the left and those on the right (which should only improve the performance).

I conducted structural disambiguation on the test data, using the probabilities like those in (6.3) and (6.4), estimated on the basis of 2D-Clustering and Brown, respectively. I also tested the method of using probabilities estimated based on word occurrences, denoted here as 'Word-based.'

Figure 6.2 shows the results in terms of accuracy and coverage, where 'coverage' refers to the percentage of test data for which the disambiguation method was able to make a decision. Since for Brown the number of word classes finally created has to be designed in advance, I tried a number of alternatives and obtained results for them (for 2D-Clustering, the optimal number of word classes is automatically selected). We see that, for Brown, when the number of word classes finally to be created is small, though the coverage will be large, the accuracy will deteriorate dramatically, indicating that in word clustering it is preferable to introduce a mechanism to automatically determine the final number of word classes.

Table 6.1 shows final results for the above methods combined with 'Default' in which we attach the first noun to the neighboring noun when a decision cannot be made by an individual method.

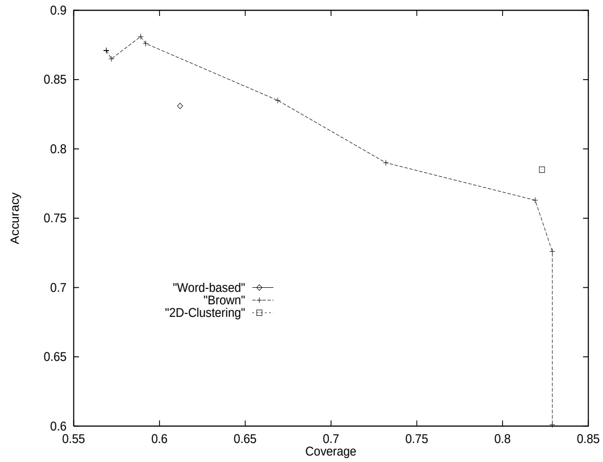


Figure 6.2: Accuracy-coverage plots for 2D-Clustering, Brown, and Word-based.

Table 6.1: Compound noun disambiguation results.

Method	Accuracy(%)
Default	59.2
Word-based + Default	73.9
Brown + Default	77.3
2D-Clustering + Default	78.3

We can see here that 2D-Clustering performs the best. These results demonstrate one desirable aspect of 2D-Clustering: its ability to *automatically* select the most appropriate level of clustering, i.e., it results in neither over-generalization nor under-generalization. (The final result of 2D-Clustering is still not completely satisfactory, however. I think that this is partly due to insufficient training data.)

6.4.3 Experiment 3: pp-attachment disambiguation

I tested the effectiveness of the proposed method by using the acquired classes in pp-attachment disambiguation involving quadruples (v, n_1, p, n_2).

As described in Chapter 3, in disambiguation of (eat, ice-cream, with, spoon), we can perform disambiguation by comparing the probabilities

$$\hat{P}_{\text{with}}(\text{spoon}|\text{eat}), \tag{6.5}$$

$$\hat{P}_{\text{with}}(\text{spoon}|\text{ice\_cream}). \tag{6.6}$$

If the former is larger, we attach 'with spoon' to 'eat;' if the latter is larger we attach it to 'ice-cream;' otherwise, we make no decision.

I used the ten sets used in Experiment 2 in Chapter 4, and conducted experiments through 'ten-fold cross validation,' i.e., all of the experimental results reported below were obtained from averages taken over ten trials.

Table 6.2: PP-attachment disambiguation results.

Method	Coverage(%)	Accuracy(%)
Default	100	56.2
Word-based	32.3	95.6
Brown	51.3	98.3
2D-Clustering	51.3	98.3
WordNet	74.3	94.5
1D-Thesaurus	42.6	97.1

I conducted word clustering by using the method proposed in this chapter, denoted as '2D-Clustering,' and the method proposed in (Brown et al., 1992), denoted as 'Brown.' In accord with the proposal offered by (Tokunaga, Iwayama, and Tanaka, 1995), for both methods, I separately conducted clustering with respect to each of the 10 most frequently occurring prepositions (e.g., 'for,' 'with,' etc). I did not cluster words with respect to rarely occurring prepositions. I then performed disambiguation by using probabilities estimated based on 2D-Clustering and Brown. I also tested the method of using the probabilities estimated based on word co-occurrences, denoted here as 'Word-based.'

Next, rather than using the co-occurrence probabilities estimated by 2D-Clustering, I only used the noun thesauruses constructed by 2D-Clustering, and applied the method of estimating the best tree cut models within the thesauruses in order to estimate conditional probabilities like those in (6.5) and (6.6). I call this method '1D-Thesaurus.'

Table 6.2 shows the results for all these methods in terms of coverage and accuracy. It also shows the results obtained in the experiment described in Chapter2, denoted here as 'WordNet.'

I then enhanced each of these methods by using a default decision of attaching (p, n_2) to n_1 when a decision cannot be made. This is indicated as 'Default.' Table 6.3 shows the results of these experiments.

We can make a number of observations from these results. (1) 2D-Clustering achieves broader coverage than does 1D-Thesaurus. This is because, in order to estimate the probabilities for disambiguation, the former exploits more information than the latter. (2) For Brown, I show here only its best result, which happens to be the same as the result for 2D-Clustering, but in order to obtain this result I had to take the trouble of conducting a number of tests to find the best level of clustering. For

Table 6.3: PP-attachment disambiguation results.

Method	Accuracy(%)
Word-based + Default	69.5
Brown + Default	76.2
2D-Clustering + Default	76.2
WordNet + Default	82.2
1D-Thesaurus + Default	73.8

2D-Clustering, this needed to be done only once and could be done automatically. (3) 2D-Clustering outperforms WordNet in term of accuracy, but not in terms of coverage. This seems reasonable, since an automatically constructed thesaurus is more domain dependent and therefore captures the domain dependent features better, thus helping achieve higher accuracy. On the other hand, with the relatively small size of training data we had available, its coverage is smaller than that of a general purpose hand-made thesaurus. The result indicates that it makes sense to combine the use of automatically constructed thesauruses with that of a hand-made thesaurus. I will describe such a method and the experimental results with regard to it in Chapter 7.

6.5 Summary

I have described in this chapter a method of clustering words. That is a natural extension of Brown et al's method. Experimental results indicate that it is superior to theirs.

The proposed clustering algorithm, 2D-Clustering, can be used in practice so long as the data size is at the level of the current Penn Tree Bank. It is still relatively computationally demanding, however, and the important work of improving its efficiency remains to be performed.

The method proposed in this chapter is not limited to word clustering; it can be applied to other tasks in natural language processing and related fields, such as, document classification (cf., (Iwayama and Tokunaga, 1995)).

Chapter 7

Structural Disambiguation

To have good fruit you must have a healthy tree; if you have a poor tree you will have bad fruit.
- The Gospel according to Matthew

In this chapter, I propose a practical method for pp-attachment disambiguation. This method combines the use of the hard co-occurrence model with that of the tree cut model.

7.1 Procedure

Let us consider here the problem of structural disambiguation, in particular, the problem of resolving pp-attachment ambiguities involving quadruples (v, n_1, p, n_2), such as (eat, ice-cream, with, spoon).

As described in Chapter 6, we can resolve such an ambiguity by using probabilities estimated on the basis of hard co-occurrence models. I denote them as

$$\hat{P}_{\text{hcm}}(\text{spoon}|\text{eat}, \text{with}),$$

$$\hat{P}_{\text{hcm}}(\text{spoon}|\text{ice\_cream}, \text{with}).$$

Further, as described in Chapter 4, we can also resolve the ambiguity by using probabilities estimated on the basis of tree cut models *with respect to a hand-made thesaurus*, denoted as

$$\hat{P}_{\text{tcm}}(\text{spoon}|\text{eat}, \text{with}),$$

$$\hat{P}_{\text{tcm}}(\text{spoon}|\text{ice\_cream}, \text{with}).$$

Both methods are a class-based approach to disambiguation, and thus can help to handle the data sparseness problem. The former method is based on corpus data and thus can capture domain specific features and achieve higher accuracy. At the same time, since corpus data is never sufficiently large, coverage is bound to be less

than satisfactory. By way of contrast, the latter method is based on human-defined knowledge and thus can bring about broader coverage. At the same time, since the knowledge used is not domain-specific, accuracy might be expected to be less than satisfactory. Since both methods have pros and cons, it would seem be better to combine the two, and I propose here a back-off method to do so.

In disambiguation, we first use probabilities estimated based on hard co-occurrence models; if the probabilities are equal (particularly both of them are 0), we use probabilities estimated based on tree cut models with respect to a hand-made thesaurus; if the probabilities are still equal, we make a default decision. Figure 7.1 shows the procedure of this method.

Procedure:
Take (v, n_1, p, n_2) as input;
if
    $\hat{P}_{\mathrm{hcm}}(n_2|v,p) > \hat{P}_{\mathrm{hcm}}(n_2|n_1,p)$
then
    attach (p, n_2) to v;
else if
    $\hat{P}_{\mathrm{hcm}}(n_2|v,p) < \hat{P}_{\mathrm{hcm}}(n_2|n_1,p)$
then
    attach (p, n_2) to n_1;
else
    if
        $\hat{P}_{\mathrm{tcm}}(n_2|v,p) > \hat{P}_{\mathrm{tcm}}(n_2|n_1,p)$
    then
        attach (p, n_2) to v;
    else if
        $\hat{P}_{\mathrm{tcm}}(n_2|v,p) < \hat{P}_{\mathrm{tcm}}(n_2|n_1,p)$
    then
        attach (p, n_2) to n_1;
    else
        make a default decision.

Figure 7.1: The disambiguation procedure.

7.2 An Analysis System

Let us consider this disambiguation method in more general terms. The natural language analysis system that implements the method operates on the basis of two processes: a learning process and an analysis process.

During the learning process, the system takes natural language sentences as input and acquires lexical semantic knowledge. First, the POS (part-of-speech) tagging module uses a probabilistic tagger (cf., Chapter 2) to assign the most likely POS tag to each word in the input sentences. The word sense disambiguation module then employs a probabilistic model (cf., Chapter 2) to resolve word sense ambiguities. Next, the case frame extracting module employs a heuristic method (cf., Chapter 2) to extract case frame instances. Finally, the learning module acquires lexical semantic knowledge (case frame patterns) on the basis of the case frame instances.

During the analysis process, the system takes a sentence as input and outputs a most likely interpretation (or several most likely interpretations). The POS tagging module assigns the most likely tag to each word in the input sentence, as is in the case of learning. The word sense disambiguation module then resolves word sense ambiguities, as is in the case of learning. The parsing module then analyzes the sentence. When ambiguity arises, the structural disambiguation module refers to the acquired knowledge, calculates the likelihood values of the ambiguous interpretations (case frames) and selects the most likely interpretation as the analysis result.

Figure 7.2 shows an outline of the system. Note that while for simplicity the parsing process and the disambiguation process are separated into two modules, they can (and usually should) be unified into one module. Furthermore, for simplicity some other knowledge necessary for natural language analysis, e.g., a grammar, has also been omitted from the figure.

The learning module consists of two submodules: a thesaurus construction submodule, and a case slot generalization submodule. The thesaurus construction submodule employs the hard co-occurrence model to calculate probabilities. The case slot generalization submodule then employs the tree cut model to calculate probabilities.

The structural disambiguation module refers to the probabilities, and calculates likelihood for each interpretation. The likelihood values based on the hard co-occurrence model for the two interpretations of the sentence (1.1) are calculated as follows

$$L_{\text{hcm}}(1) = \hat{P}_{\text{hcm}}(\text{I}|\text{eat}, \text{arg1}) \cdot \hat{P}_{\text{hcm}}(\text{ice\_cream}|\text{eat}, \text{arg2}) \cdot \hat{P}_{\text{hcm}}(\text{spoon}|\text{eat}, \text{with})$$

$$L_{\text{hcm}}(2) = \hat{P}_{\text{hcm}}(\text{I}|\text{eat}, \text{arg1}) \cdot \hat{P}_{\text{hcm}}(\text{ice\_cream}|\text{eat}, \text{arg2}) \cdot \hat{P}_{\text{hcm}}(\text{spoon}|\text{girl}, \text{with}).$$

The likelihood values based on the tree cut model can be calculated analogously. Finally, the disambiguation module selects the most likely interpretation on the basis of a back-off procedure like that described in Section 1.

Note that in its current state of development, the disambiguation module is still unable to exploit syntactic knowledge. As described in Chapter 2, disambiguation

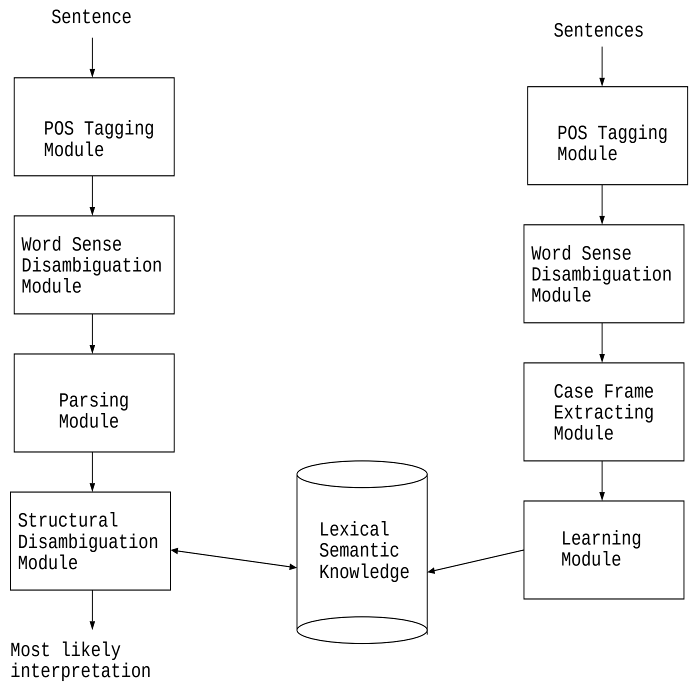


Figure 7.2: Outline of the natural language analysis system.

decisions may not be made solely on the basis of lexical knowledge; it is necessary to utilize syntactic knowledge as well. Further study is needed to determine how to define a unified model which combines both lexical knowledge and syntactic knowledge. In terms of syntactic factors, we need to consider psycholinguistic principles, e.g., the 'right association principle.' I have found in my study that using a probability model embodying these principles helps improve disambiguation results (Li, 1996). Another syntactic factor we need to take into consideration is the likelihood of the phrase structure of an interpretation (cf., (Charniak, 1997; Collins, 1997; Shirai et al., 1998)).

7.3 Experimental Results

I tested the proposed disambiguation method by using the data used in Chapters 4 and 6. Table 7.1 shows the results; here the method is denoted as '2D-Clustering+WordNet+Default.' Table 7.1 also shows the results of WordNet+Default and TEL which were described in Chapter 4, and the result of 2D-Clustering+Default which was described in Chapter 6. We see that the disambiguation method proposed in this chapter performs the best of four.

Table 7.2 shows the disambiguation results reported in other studies. Since the

Table 7.1: PP-attachment disambiguation results.

TEL	82.4
2D-Clustering + Default	76.2
WordNet + Default	82.2
2D-Clustering + WordNet + Default	85.2

data sets used in the respective studies were different, a straightforward comparison of the various results would have little significance, we may say that the method proposed in this chapter appears to perform relatively well with respect to other state-of-the-art methods.

Table 7.2: Results reported in previous work.

Method	Data	Accuracy (%)
(Hindle and Rooth, 1991)	AP News	78.3
(Resnik, 1993a)	WSJ	82.2
(Brill and Resnik, 1994)	WSJ	81.8
(Ratnaparkhi, Reynar, and Roukos, 1994)	WSJ	81.6
(Collins and Brooks, 1995)	WSJ	84.5

Chapter 8

Conclusions

If all I know is a fraction, then my only fear is of losing the thread.

- Lao Tzu

8.1 Summary

The problem of acquiring lexical semantic knowledge is an important issue in natural language processing, especially with regard to structural disambiguation. The approach I have adopted here to this problem has the following characteristics: (1) dividing the problem into three subproblems: case slot generalization, case dependency learning, and word clustering, (2) viewing each subproblem as that of statistical estimation and defining probability models (probability distributions) for each subproblem, (3) adopting MDL as a learning strategy, (4) employing efficient learning algorithms, and (5) viewing the disambiguation problem as that of statistical prediction.

Major contributions of this thesis include: (1) formalization of the lexical knowledge acquisition problem, (2) development of a number of learning methods for lexical knowledge acquisition, and (3) development of a high-performance disambiguation method.

Table 8.1 shows the models I have proposed, and Table 8.2 shows the algorithms I have employed. The overall accuracy achieved by the pp-attachment disambiguation method is 85.2%, which is better than that of state-of-the-art methods.

8.2 Open Problems

Lexical semantic knowledge acquisition and structural disambiguation are difficult tasks. Although I think that the investigations reported in this thesis represent some significant progress, further research on this problem is clearly still needed.

Table 8.1: Models proposed.

Purpose	Basic model	Restricted model
case slot generalization	case slot model (hard, soft)	tree cut model
case dependency learning	case frame model (word-based, class-based, slot-based)	dependency forest model
word clustering	co-occurrence model (hard, soft)	hard co-occurrence model

Table 8.2: Algorithm employed.

Purpose	Algorithm	Time complexity
case slot generalization	Find-MDL	$O(N)$
case dependency learning	Suzuki's algorithm	$O(n^2(k^2+n^2))$
word clustering	2D-Clustering	$O(N^3 \cdot V + V^3 \cdot N)$

Other issues not investigated in this thesis and some possible solutions include:

More complicated models In the discussions so far, I have restricted the class of hard case slot models to that of tree cut models for an existing thesaurus *tree*. Under this restriction, we can employ an efficient dynamic-programming-based learning algorithm which can provablely find the optimal MDL model. In practice, however, the structure of a thesaurus may be a directed acyclic graph (DAG) and straightforwardly extending the algorithm to a DAG may no longer guarantee that the optimal model will be found. The question now is whether there exist sub-optimal algorithms for more complicated model classes. The same problem arises in case dependency learning, for which I have restricted the class of case frame models to that of dependency forest models. It would be more appropriate, however, to restrict the class to, for example, the class of normal Bayesian Networks. How to learn such a complicated model, then, needs further investigation.

Unified model I have divided the problem of learning lexical knowledge into three subproblems for easy examination. It would be more appropriate to define a single unified model. How to define such a model, as well as how to learn it, are issues for future research. (See (Miyata, Utsuro, and Matsumoto, 1997; Utsuro and Matsumoto, 1997) for some recent progress on this issue; see also discussions in Chapter 3.)

Combination with extraction We have seen that the amount of data currently available is generally far less than that necessary for accurate learning, and the problem of how to collect sufficient data may be expected to continue to be a crucial issue. One solution might be to employ bootstrapping, i.e., to conduct extraction and generalization, iteratively. How to combine the two processes needs further examination.

Combination with word sense disambiguation I have not addressed the word sense ambiguity problem in this thesis, simply proposing to conduct word sense disambiguation in pre-processing. (See (McCarthy, 1997) for her proposal on word sense disambiguation.) In order to improve the disambiguation results, however, it would be better to employ the soft case slot model to perform structural and word sense disambiguation at the same time. How to effectively learn such a model requires further work.

Soft clustering I have formalized the problem of constructing a thesaurus into that of learning a double mixture model. How to efficiently learn such a model is still an open problem.

Parsing model The use of lexical knowledge alone in disambiguation might result in the resolving of most of the ambiguities in sentence parsing, but not all of them. As has been described, one solution to the problem might be to define a unified model combining both lexical knowledge and syntactic knowledge. The problem still requires further work.

References

[Abe and Li1996] Abe, Naoki and Hang Li. 1996. Learning word association norms using tree cut pair models. *Proceedings of the 13th International Conference on Machine Learning*, pages 3–11.

[Abe, Li, and Nakamura1995] Abe, Naoki, Hang Li, and Atsuyoshi Nakamura. 1995. On-line learning of binary lexical relations using two-dimensional weighted majority algorithms. *Proceedings of the 12th International Conference on Machine Learning*, pages 3–11.

[Almuallim et al.1994] Almuallim, Hussein, Yasuhiro Akiba, Takefumi Yamazaki, Akio Yokoo, and Shigeo Kaneda. 1994. Two methods for ALT-J/E translation rules from examples and a semantic hierarchy. *Proceedings of the 15th International Conference on Computational Linguistics*, pages 57–63.

[Alshawi and Carter1994] Alshawi, Hiyan and David Carter. 1994. Training and scaling preference functions for disambiguation. *Computational Linguistics*, 20(4):635–648.

[Altmann and Steedman1988] Altmann, Gerry and Mark Steedman. 1988. Interaction with context during human sentence processing. *Cognition*, 30:191–238.

[Bahl, Jelinek, and Mercer1983] Bahl, Lalit R., Frederick Jelinek, and Robert Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 5(2):179–190.

[Barron and Cover1991] Barron, Andrew R. and Thomas M. Cover. 1991. Minimum complexity density estimation. *IEEE Transaction on Information Theory*, 37(4):1034–1054.

[Berger, Pietra, and Pietra1996] Berger, Adam L., Stephen J. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

[Black1988] Black, Ezra. 1988. An experiment in computational discrimination of English word senses. *IBM Journal of Research and Development*, 32(2):185–193.

[Black et al.1993] Black, Ezra, Fred Jelinek, John Lafferty, and David M. Magerman. 1993. Towards history-based grammars: Using richer models for probabilistic parsing. *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 31–37.

[Brent1991] Brent, Michael R. 1991. Automatic acquisition of subcategorization frames from untagged text. *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 209–214.

[Brent1993] Brent, Michael R. 1993. From grammar to lexicon: Unsupervised learning of lexical syntax. *Computational Linguistics*, 19(2):243–262.

[Brent and Cartwright1996] Brent, Michael R. and Timothy A. Cartwright. 1996. Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61:93–125.

[Brent, Murthy, and Lundberg1995] Brent, Michael R., Sreerama K. Murthy, and Andrew Lundberg. 1995. Discovering morphemic suffixes: A case study in minimum description length induction. *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*.

[Brill1995] Brill, Eric. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

[Brill and Resnik1994] Brill, Eric and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. *Proceedings of the 15th International Conference on Computational Linguistics*, pages 1198–1204.

[Briscoe and Carroll1993] Briscoe, Ted and John Carroll. 1993. Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59.

[Briscoe and Carroll1997] Briscoe, Ted and John Carroll. 1997. Automatic extraction of subcategorization from corpora. *Proceedings of the 5th Conference on Applied Natural Language Processing*.

[Brown et al.1991] Brown, Peter, Stephen Della Pietra, Vincent Della Pietra, and Robert Mercer. 1991. Word sense disambiguation using statistical methods. *Proceedings of Annual the 29th Meeting of the Association for Computational Linguistics*, pages 264–270.

[Brown et al.1992] Brown, Peter F., Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):283–298.

[Bruce and Wiebe1994] Bruce, Rebecca and Janyce Wiebe. 1994. Word-sense disambiguation using decomposable models. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 139–145.

[Carter1997] Carter, David. 1997. The treebanker: a tool for supervised training of parsed corpora. *Proceedings of the Computational Environments for Grammar Development*.

[Cartwright and Brent1994] Cartwright, Timothy A. and Michael R. Brent. 1994. Segmenting speech without a lexicon: The roles of phonotactics and speech source. *Proceedings of the 1st Meeting of the ACL Special Interest Group in Computational Phonology*, pages 83–90.

[Chang, Luo, and Su1992] Chang, Jing-Shin, Yih-Fen Luo, and Keh-Yih Su. 1992. GPSM: A generalized probabilistic semantic model for ambiguity resolution. *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 177–184.

[Charniak1997] Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. *Proceedings of the 15th National Conference on Artificial Intelligency.*

[Charniak et al.1993] Charniak, Eugene, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. 1993. Equations for part-of-speech tagging. *Proceedings of the 11th National Conference on Artificial Intelligency*, pages 784–789.

[Chen and Chen1994] Chen, Kuang-hua and Hisn-Hsi Chen. 1994. Extracting noun phrases from large-scale texts: A hybrid approach and its automatic evaluation. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 234–241.

[Chiang, Lin, and Su1995] Chiang, Tung-Hui, Yi-Chung Lin, and Keh-Yih Su. 1995. Robust learing, smoothing, and parameter tying on syntactic ambiguity resolution. *Computational Linguistics*, 21(3):321–349.

[Chitrao and Grishman1990] Chitrao, Mahesh V. and Ralph Grishman. 1990. Statistical parsing of messages. *Proceedings of DARPA Speech and Natural Language Workshop*, pages 263–266.

[Chow and Liu1968] Chow, C.K. and C.N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

[Church et al.1989] Church, Kenneth, William Gale, Patrick Hanks, and Donald Hindle. 1989. Parsing, word associations and typical predicate-argument relations. *Proceedings of the International Workshop on Parsing Technology '89*, pages 389–398.

[Church et al.1991] Church, Kenneth, William Gale, Patrick Hanks, and Donald Hindle. 1991. Using statistics in lexical analysis. In Uri Zernik, editor, *Lexical Acquisition: Exploiting On-line Resources to Build a Lexicon*. Lawrence ErlBaum Associates, Hillsdale, pages 115–164.

[Church and Hanks1989] Church, Kenneth and Patrick Hanks. 1989. Word association norms, mutual information, and lexicography. *Proceedings of the 27th Annual Meeting of Association for Computational Linguistics*, pages 76–83.

[Church1988] Church, Kenneth. W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. *Proceedings of the 2nd Conference on Applied Natural Language Processing*, pages 136–143.

[Clarke and Barron1990] Clarke, Bertrand S. and Andrew R. Barron. 1990. Information-theoretic asymptotics of Bayes methods. *IEEE Transaction on Information Theory*, 36(3):453–471.

[Collins1996] Collins, Michael. 1996. A new statistical parser based on bigram lexical dependencies. *Proceedings of the 34th Annual Meeting of the Association for Computation al Linguistics*, pages 184–191.

[Collins1997] Collins, Michael. 1997. Three generative lexicalized models for statistical parsing. *Proceedings of the 35th Annual Meeting of the Association for Computation al Linguistics*, pages 16–23.

[Collins and Brooks1995] Collins, Michael and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. *Proceedings of the 3rd Workshop on Very Large Corpora*.

[Cooper and Herskovits1992] Cooper, Gregory F. and Edward Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

[Cover and Thomas1991] Cover, Thomas M. and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons Inc., New York.

[Cucchiarelli and Velardi1997] Cucchiarelli, Alessandro and Paola Velardi. 1997. Automatic selection of class labels from a thesaurus for an effective semantic tagging of corpora. *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 380–387.

[Dagan, Marcus, and Makovitch1992] Dagan, Ido, Shaul Marcus, and Shaul Makovitch. 1992. Contextual word similarity and estimation from sparse data. *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 164–171.

[Dagan, Pereira, and Lee1994] Dagan, Ido, Fernando Pereira, and Lillian Lee. 1994. Similarity-based estimation of word cooccurrence probabilities. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 272–278.

[Dagan, Pereira, and Lee1997] Dagan, Ido, Fernando Pereira, and Lillian Lee. 1997. Similarity-based methods for word sense disambiguation. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 56–63.

[Darroch and Ratcliff1972] Darroch, J. N. and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(5):1740–1480.

[Dempster, Laird, and Rubin1977] Dempster, A.P., N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

[Den1996] Den, Yasuharu. 1996. *A Uniform Approach ot Spoken Language Analysis*. Ph.D. Thesis, Kyoto Univ.

[Ellison1991] Ellison, T. Mark. 1991. Discovering planar segregations. *Proceedings of AAAI Spring Symposium on Machine Learning of Natural Language and Ontology*, pages 42–47.

[Ellison1992] Ellison, T. Mark. 1992. Discovering vowel harmony. In Walter Daelmans and David Powers, editors, *Background and Experiments in Machine Learning of Natural Language*. pages 205–207.

[Fillmore1968] Fillmore, C. 1968. The case for case. In E. Bach and R. Harms, editors, *Universals in Linguistics theory*. Holt, RineHart, and Winston, New York.

[Fisher1956] Fisher, R.A. 1956. *Statistical Methods and Scientific Inference*. Olyver and Boyd.

[Ford, Bresnan, and Kaplan1982] Ford, Marylyn, Joan Bresnan, and Ronald Kaplan. 1982. A competence based theory of syntactic closure. *The Mental Representation of Grammatical Relations*.

[Framis1994] Framis, Francesc Ribas. 1994. An experiment on learning appropriate selectional restrictions from a parsed corpus. *Proceedings of the 15th International Conference on Computational Linguistics*, pages 769–774.

[Frazier and Fodor1979] Frazier, Lyn and Janet Fodor. 1979. The sausage machine: A new two-stage parsing model. *Cognition*, 6:291–325.

[Fujii et al.1996] Fujii, Atsushi, Kentaro Inui, Takenobu Tokunaga, and Hozumi Tanaka. 1996. To what extent does case contribute to verb sense disambiguation. *Proceedings of the 16th International Conference on Computational Linguistics*, pages 59–64.

[Fujisaki et al.1989] Fujisaki, T., F. Jelinek, J. Cocke, E. Black, and T. Nishino. 1989. A probabilistic parsing method for sentence disambiguation. *Proceedings of the International Workshop on Parsing Technology '89*, pages 85–94.

[Gale, Church, and Yarowsky1992] Gale, William, Kenneth Ward Church, and David Yarowsky. 1992. Estimating upper and lower bounds on the performance of word-sense disambiguation programs. *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 249–256.

[Gale and Church1990] Gale, Williams A. and Kenneth W. Church. 1990. Poor estimates of context are worse than none. *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 283–287.

[Gelfand and Smith1990] Gelfand, Alan E. and Adrian F.M. Smith. 1990. Sampling-based approach to calculating marginal densities. *Journal of the American Statistical Association*, 85(40):398–409.

[Geman and Geman1984] Geman, Stuart and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions and the Bayes restoration of images. *IEEE Tran. on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.

[Golding and Roth1996] Golding, Andrew R. and Dan Roth. 1996. Applying winnow to context-sensitive spelling correction. *Proceedings of the 13th International Conference on Machine Learning*, pages 182–190.

[Golding and Schabes1996] Golding, Andrew R. and Yves Schabes. 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 71–78.

[Grefenstette1994] Grefenstette, Gregory. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Boston.

[Grishman and Sterling1992] Grishman, Ralph and John Sterling. 1992. Acquisition of selectional patterns. *Proceedings of the 14th International Conference on Computational Linguistics*, pages 658–664.

[Grishman and Sterling1994] Grishman, Ralph and John Sterling. 1994. Generalizing automatically generated selectional patterns. *Proceedings of the 15th International Conference on Computational Linguistics*, pages 742–747.

[Grunwald1996] Grunwald, Peter. 1996. A minimum description length approach to grammar inference. In S. Wemter, E. Riloff, and G. Scheler, editors, *Symbolic, Connectionist and Statistical Approaches to Learning for Natural Language Processing, Lecture Note in AI*. Springer Verlag, pages 203–216.

[Guthrie et al.1991] Guthrie, Joe A., Louise Guthrie, Yorick Wilks, and Homa Aidinejad. 1991. Subject-dependent co-occurrence and word sense disambiguation. *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 146–152.

[Han and Kobayashi1994] Han, Te Sun and Kingo Kobayashi. 1994. *Mathematics of Information and Coding*. Iwanami Shoten Publishers, Tokyo.

[Haruno and Matsumoto1997] Haruno, Masahiko and Yuji Matsumoto. 1997. Mistake-driven mixture of hierarchical tag context trees. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 230–237.

[Haruno, Shirai, and Ooyama1998] Haruno, Masahiko, Satoshi Shirai, and Yoshifumi Ooyama. 1998. Using decision trees to construct a practical parser. *Machine Learning (to appear)*.

[Hastings1970] Hastings, W.K. 1970. Monte Carlo sampling method using Markov chains and their applications. *Biometrika*, 57:97–109.

[Helmbold et al.1995] Helmbold, David P., Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. 1995. A comparison of new and old algorithm for a mixture estimation problem. *Proceedings of the 8th Annual Conference on Computational Learning Theory*, pages 69–78.

[Hindle1990] Hindle, Donald. 1990. Noun classification from predicate-argument structures. *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 268–275.

[Hindle and Rooth1991] Hindle, Donald and Mats Rooth. 1991. Structural ambiguity and lexical relations. *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 229–236.

[Hindle and Rooth1993] Hindle, Donald and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

[Hobbs and Bear1990] Hobbs, Jerry R. and John Bear. 1990. Two principles of parse preference. *Proceedings of the 13th International Conference on Computational Linguistics*, pages 162–167.

[Hogenhout and Matsumoto1996] Hogenhout, Wide R. and Yuji Matsumoto. 1996. Training stochastical grammars on semantical categories. In Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*. Springer.

[Hogenhout and Matsumoto1997] Hogenhout, Wide R. and Yuji Matsumoto. 1997. A preliminary study of word clustering based on syntactic behavior. *CoNLL97:Computational Natural Language Learning, Proceedings of the 1997 Meeting of the ACL Special Interest Group in Natural Language Learning*, pages 16–24.

[Inui, Sornlertlamvanich, and Tanaka1998] Inui, Kentaro, Virach Sornlertlamvanich, and Hozumi Tanaka. 1998. Probabilistic GLR parsing: A new formalization and its impact on parsing performance. *Journal of Natural Language Processing*, 5(3):33–52.

[Iwayama and Tokunaga1995] Iwayama, Makoto and Takenobu Tokunaga. 1995. Cluster-based text categorization: A comparison of category search strategy. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 273–280.

[Jaynes1978] Jaynes, E. T. 1978. Where do we stand on maximum entropy? In R. D. Levine and M. Tribus, editors, *The Maximum Entropy Formalism*. MIT Press.

[Jeffreys1961] Jeffreys, H. 1961. *Theory of Probability*. Oxford Univ. Press.

[Jelinek, Lafferty, and Mercer1990] Jelinek, F., J. D. Lafferty, and R. L. Mercer. 1990. Basic methods of probabilistic context free grammars. *IBM Research Report, RC 16374*.

[Jelinek and Mercer1980] Jelinek, F. and R. I. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In E.S. Gelseman and L.N. Kanal, editors, *Pattern Recognition in Practice*. North-Holland Publishing Company, pages 381–397.

[Johnson-Laird1983] Johnson-Laird, P. N. 1983. *Mental Model: Towards a Cognitive Science of Language, Inference, and Consciousness*. Harvard Univ. Press.

[Katz and Fodor1963] Katz, J. J. and J. A. Fodor. 1963. The structure of semantic theory. *Language*, 39:170–210.

[Katz1987] Katz, Slava M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(3):400–401.

[Kimball1973] Kimball, John. 1973. Seven principles of surface structure parsing in natural language. *Cognition*, 2(1):15–47.

[Kirkpatrick, Gelatt, and Vecchi1983] Kirkpatrick, S., C. D. Gelatt, and M.P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220:671–680.

[Kita1992] Kita, Kenji. 1992. *A Study on Language Modeling for Speech Recognition*. Ph.D Thesis, Waseda Univ.

[Knuth1973] Knuth, Donald E. 1973. *The Art of Computer Programming (Second Edition)*. Addison-Wesley Publishing Company.

[Krichevskii and Trofimov1981] Krichevskii, R. E. and V. K. Trofimov. 1981. The performance of universal coding. *IEEE Transaction on Information Theory*, 27(2):199–207.

[Kupiec1992] Kupiec, Julian M. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225–242.

[Kurohashi and Nagao1994] Kurohashi, Sadao and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

[Lakoff1987] Lakoff, George. 1987. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. The University of Chicago Press.

[Lari and Young1990] Lari, K. and S.J. Young. 1990. The estimation of stochastic context free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

[Leacock, Towell, and Voorhees1993] Leacock, Claudia, Geoffrey Towell, and Ellen Voorhees. 1993. Corpus-based statistical sense resolution. *Proceedings of ARPA Workshop on Human Language Technology*.

[Li1996] Li, Hang. 1996. A probabilistic disambiguation method based on psycholinguistic principles. *Proceedings of the 4th Workshop on Very Large Corpora*, pages 141–154.

[Li and Abe1996] Li, Hang and Naoki Abe. 1996. Clustering words with the MDL principle. *Proceedings of the 16th International Conference on Computational Linguistics*, pages 4–9.

[Li and Abe1997] Li, Hang and Naoki Abe. 1997. Clustering words with the MDL principle. *Journal of Natural Language Processing*, 4(2):71–88.

[Li and Yamanishi1997] Li, Hang and Kenji Yamanishi. 1997. Document classification using a finite mixture model. *Proceedings of the 35th Annual Meeting of Association for Computational Linguistics*, pages 71–88.

[Littlestone1988] Littlestone, Nick. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318.

[Littlestone and Warmuth1994] Littlestone, Nick and Manfred K. Warmuth. 1994. The weighted majority algorithm. *Information and Computation*, 108:212–261.

[Magerman1994] Magerman, David M. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. Thesis, Stanford Univ.

[Magerman1995] Magerman, David M. 1995. Statistical decision-tree models for parsing. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283.

[Magerman and Marcus1991] Magerman, David M. and Mitchell P. Marcus. 1991. Pearl: A probabilistic chart parser. *Proceedings of the International Workshop on Parsing Technology '91*, pages 193–199.

[Manning1992] Manning, Christopher D. 1992. Automatic acquisition of a large subcategorization dictionary from corpora. *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 235–242.

[Marcus, Santorini, and Marcinkiewicz1993] Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(1):313–330.

[Matsumoto, Ishimoto, and Utsuro1993] Matsumoto, Yuji, Hiroyuki Ishimoto, and Takehito Utsuro. 1993. Structural matching of parallel texts. *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 23–30.

[McCarthy1997] McCarthy, Diana. 1997. Word sense disambiguation for acquisition of selectional preferences. *Proceedings of ACL/EACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 52–60.

[McKeown and Hatzivassiloglou1993] McKeown, Kathleen and Vasileios Hatzivassiloglou. 1993. Augmenting lexicons automatically: Clustering semantically related adjectives. *Proceedings of ARPA Workshop on Human Language Technology.*

[McMahon and Smith1996] McMahon, John C. and Francis J. Smith. 1996. Improving statistical language model performance with automatically generated word hierarchies. *Computational Linguistics*, 22(2):217–247.

[McRoy1992] McRoy, Susan W. 1992. Using multiple knowledge sources for word sense discrimination. *Computational Linguistics*, 18(1):1–30.

[Merialdo1994] Merialdo, Bernard. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

[Miller1995] Miller, George A. 1995. WordNet: A lexical database for English. *Communications of the ACM*, pages 39–41.

[Miyata, Utsuro, and Matsumoto1997] Miyata, Takashi, Takehito Utsuro, and Yuji Matsumoto. 1997. Bayesian network models of subcategorization and their MDL-based learning from corpus. *Proceedings of the 4th Natural Language Processing Pacific Rim Symposium*, pages 321–326.

[Nagao1990] Nagao, Katashi. 1990. Dependency analyzer: A knowledge-based approach to structural disambiguation. *Proceedings of the 13th International Conference on Computational Linguistics*, 2:282–287.

[Nagata1994] Nagata, Masaaki. 1994. A stochastic Japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm. *Proceedings of the 15th International Conference on Computational Linguistics*, pages 201–207.

[Ng and Lee1996] Ng, Hwee Tou and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An examplar-based approach. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 40–47.

[Niwa and Nitta1994] Niwa, Yoshiki and Yoshihiko Nitta. 1994. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. *Proceedings of the 14th International Conference on Computational Linguistics*, pages 304–309.

[Nomiyama1992] Nomiyama, Hiroshi. 1992. Machine translation by case generalization. *Proceedings of the 14th International Conference on Computational Linguistics*, pages 714–720.

[Pearl1988] Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers Inc., San Mateo, California.

[Pereira and Singer1995] Pereira, Fernando and Yoram Singer. 1995. Beyond word n-grams. *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 183–190.

[Pereira and Tishby1992] Pereira, Fernando and Naftali Tishby. 1992. Distributional clustering, phase transitions and hierarchical clustering. *Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 108–112.

[Pereira, Tishby, and Lee1993] Pereira, Fernando, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

[Pollard and Sag1987] Pollard, C. and I. A. Sag. 1987. *Information-Based Syntax and Semantics. Volume 1: Syntax. CSLI Lecture Notes 13.* Chicago Univ. Press.

[Quinlan and Rivest1989] Quinlan, J. Ross and Ronald L. Rivest. 1989. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248.

[Ratnaparkhi1996] Ratnaparkhi, Adwait. 1996. A maximum entroy part-of-speech tagger. *Proceedings of 1st Conference on Empirical Methods in Natural Language Processing.*

[Ratnaparkhi1997] Ratnaparkhi, Adwait. 1997. A linear observed time statistical parser based on maximum models. *Proceedings of 2nd Conference on Empirical Methods in Natural Language Processing*.

[Ratnaparkhi, Reynar, and Roukos1994] Ratnaparkhi, Adwait, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. *Proceedings of ARPA Workshop on Human Language Technology*, pages 250–255.

[Resnik1992] Resnik, Philip. 1992. WordNet and distributional analysis: A class-based approach to lexical discovery. *Proceedings of AAAI Workshop on Statistically-based NLP Techniques*.

[Resnik1993a] Resnik, Philip. 1993a. *Selection and Information: A Class-based Approach to Lexical Relationships*. Ph.D. Thesis, Univ. of Pennsylvania.

[Resnik1993b] Resnik, Philip. 1993b. Semantic classes and syntactic ambiguity. *Proceedings of ARPA Workshop on Human Language Technology*.

[Ribas1995] Ribas, Francesc. 1995. On learning more appropriate selectional restrictions. *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*.

[Rissanen1983] Rissanen, Jorma. 1983. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431.

[Rissanen1984] Rissanen, Jorma. 1984. Universal coding, information, predication and estimation. *IEEE Transaction on Information Theory*, 30(4):629–636.

[Rissanen1989] Rissanen, Jorma. 1989. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Co., Singapore.

[Rissanen1996] Rissanen, Jorma. 1996. Fisher information and stochastic complexity. *IEEE Transaction on Information Theory*, 42(1):40–47.

[Rissanen1997] Rissanen, Jorma. 1997. Stochastic complexity in learning. *Journal of Computer and System Sciences*, 55:89–95.

[Ristad and Thomas1995] Ristad, Eric Sven and Robert G. Thomas. 1995. New techniques for context modeling. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.

[Rivest1987] Rivest, Ronald L. 1987. Learning decision lists. *Machine Learning*, pages 229–246.

[Rose, Gurewitz, and Fox1990] Rose, Kenneth, Eitan Gurewitz, and Geoffrey C. Fox. 1990. Statistical mechanics and phase transitions in clustering. *Physical Review Letters*, 65(8):945–948.

[Rosenfeld1996] Rosenfeld, Ronald. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computational Linguistics*, 22(1):39–71.

[Samuelsson1995] Samuelsson, Christer. 1995. A novel framework for reductionistic statistical parsing. *Proceedings of International Workshop on Parsing Technology '95*, pages 208–215.

[Saul and Pereira1997] Saul, Lawrence and Fernando Pereira. 1997. Aggregate and mixed-order Markov models for statistical language processing. *Proceedings of 2nd Conference on Empirical Methods in Natural Language Processing*.

[Schabes1992] Schabes, Yves. 1992. Stochastic lexicalized tree-adjoining grammars. *Proceedings of the 14th International Conference on Computational Linguistics*, pages 425–432.

[Schütze1997] Schütze, Hinrich. 1997. *Ambiguity Resolution in Language Learning: Computational and Cognitive Models*. CSLI Stanford.

[Schütze1998] Schütze, Hinrich. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

[Schütze and Singer1994] Schütze, Hinrich and Yoram Singer. 1994. Part-of-speech tagging using a variable memory Markov model. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 181–187.

[Schwarz1978] Schwarz, G. 1978. Estimation of the dimension of a model. *Annals of Statistics*, 6:416–446.

[Sekine et al.1992] Sekine, Satoshi, Jeremy J. Carroll, Sofia Ananiadou, and Jun'ichi Tsujii. 1992. Automatic learning for semantic collocation. *Proceedings of the 3rd Conference on Applied Natural Language Processing*, pages 104–110.

[Shirai et al.1998] Shirai, Kiyoaki, Kentaro Inui, Takenobu Tokunaga, and Hozumi Tanaka. 1998. A framework of intergrating syntactic and lexical statistics in statistical parsing. *Journal of Natural Language Processing*, 5(3):85–106.

[Smadja1993] Smadja, Frank. 1993. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1):143–177.

[Solomonoff1964] Solomonoff, R.J. 1964. A formal theory of inductive inference 1 and 2. *Information and Control*, 7:1–22,224–254.

[Stolcke and Omohundro1994] Stolcke, Andreas and Stephen Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In Rafael C. Carrasco and Jose Oncina, editors, *Grammatical Inference and Applications*. Springer Verlag, pages 106–118.

[Stolcke and Segal1994] Stolcke, Andreas and Jonathan Segal. 1994. Precise n-gram probabilities from stochastic context-free grammars. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 74–79.

[Su and Chang1988] Su, Keh-Yih and Jing-Shin Chang. 1988. Semantic and syntactic aspects of score function. *Proceedings of the 12th International Conference on Computational Linguistics*, pages 642–644.

[Su et al.1989] Su, Keh-Yih, Jong-Nae Wang, Mei-Hui Su, and Jin-Shin Chang. 1989. A sequential truncation parsing algorithm based on the score function. *Proceedings of the International Workshop on Parsing Technology '89*, pages 95–104.

[Suzuki1993] Suzuki, Joe. 1993. A construction of Bayesian networks from databases based on an MDL principle. *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 266–273.

[Tanaka1994] Tanaka, Hideki. 1994. Verbal case frame acquisition from a bilingual corpus: Gradual knowledge acquisition. *Proceedings of the 15th International Conference on Computational Linguistics*, pages 727–731.

[Tanaka1996] Tanaka, Hideki. 1996. Decision tree learning algorithm with structured attributes: Application to verbal case frame acquisition. *Proceedings of the 16th International Conference on Computational Linguistics*, pages 943–948.

[Tanner and Wong1987] Tanner, Martin A. and Wing Hung Wong. 1987. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398):528–540.

[Tokunaga, Iwayama, and Tanaka1995] Tokunaga, Takenobu, Makoto Iwayama, and Hozumi Tanaka. 1995. Automatic thesaurus construction based-on grammatical relations. *Proceedings of the 14th International Joint Conference on Artificial Intelligency*, pages 1308–1313.

[Tsujii1987] Tsujii, Jun'ichi. 1987. *Knowledge Representation and Use – From AI View Point (in Japanese).* Shoukoudou,Tokyo.

[Ueda and Nakano1998] Ueda, Naonori and Ryohei Nakano. 1998. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282.

[Ushioda1996] Ushioda, Akira. 1996. Hierarchical clustering of words and application to NLP tasks. *Proceedings of the 4th Workshop on Very Large Corpora*, pages 28–41.

[Utsuro and Matsumoto1997] Utsuro, Takehito and Yuji Matsumoto. 1997. Learning probabilistic subcategorization preference by identifying case dependencies and optimal noun class generalization level. *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 364–371.

[Utsuro, Matsumoto, and Nagao1992] Utsuro, Takehito, Yuji Matsumoto, and Makoto Nagao. 1992. Lexical knowledge acquisition from bilingual corpora. *Proceedings of the 14th International Conference on Computational Linguistics*, pages 581–587.

[Velardi, Pazienza, and Fasolo1991] Velardi, Paola, Maria Teresa Pazienza, and Michela Fasolo. 1991. How to encode semantic knowledge: A method for meaning representation and computer-aided acquisition. *Computational Linguistics*, 17(2):153–170.

[Voorhees, Leacock, and Towell1995] Voorhees, Ellen M., Claudia Leacock, and Geoffrey Towell. 1995. Learning context to disambiguate word senses. In T. Petsche, S.J.Hanson, and J.W. Shavlik, editors, *Computational Learning Theory and Natural Language Learning Systems 3: Selecting Good Models*. MIT Press, pages 279–305.

[Wallace and Boulton1968] Wallace, C. and D. M. Boulton. 1968. An information measure for classification. *Computer Journal*, 11:185–195.

[Webster and Marcus1989] Webster, Mort and Mitch Marcus. 1989. Automatic acquisition of the lexical semantics of verbs from sentence frames. *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 177–184.

[Wermter1989] Wermter, Stefan. 1989. Integration of semantic and syntactic constraints for structural noun phrase disambiguation. *Proceedings of the 11th International Joint Conference on Artificial Intelligency*, pages 1486–1491.

[Whittemore, Ferrara, and Brunner1990] Whittemore, Greg, Kathleen Ferrara, and Hans Brunner. 1990. Empirical study of predictive powers of simple attachment schemes for post-modifier prepositional phrases. *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 23–30.

[Wilks1975] Wilks, Yorick. 1975. An intelligent analyzer and understander of English. *Communications of the ACM*, 18(5):264–274.

[Wright1990] Wright, J.H. 1990. Lr parsing of probabilistic grammars with input uncertainty for speech recognition. *Computer Speech and Language*, 4:297–323.

[Yamanishi1992a] Yamanishi, Kenji. 1992a. A learning criterion for stochastic rules. *Machine Learning*, 9:165–203.

[Yamanishi1992b] Yamanishi, Kenji. 1992b. *A Statistical Approach to Computational Learning Theory*. Ph.D. Thesis, Univ. of Tokyo.

[Yamanishi1996] Yamanishi, Kenji. 1996. A randomized approximation of the MDL for stochastic models with hidden variables. *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 99–109.

[Yarowsky1992] Yarowsky, David. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. *Proceedings of the 14th International Conference on Computational Linguistics*, pages 454–460.

[Yarowsky1993] Yarowsky, David. 1993. One sense per collocation. *Proceedings of ARPA Workshop on Human Language Technology.*

[Yarowsky1994] Yarowsky, David. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 88–95.

[Yarowsky1995] Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196.

Appendix A

A.1 Derivation of Description Length: Two-stage Code

We consider here

$$\min_\delta \left(\log \tfrac{V}{\delta_1 \cdots \delta_k} - \log P_{\tilde{\theta}}(x^n)\right). \tag{A.1}$$

We first make Taylor's expansion of $-\log P_{\tilde{\theta}}(x^n)$ around $\hat{\theta}$:

$$\begin{array}{l} -\log P_{\tilde{\theta}}(x^n) = -\log P_{\hat{\theta}}(x^n) + \frac{\partial(-\log P_\theta(x^n))}{\partial \theta}|_{\hat{\theta}} \cdot \delta \\ +\frac{1}{2} \cdot \delta^T \cdot \left\{ \frac{\partial^2(-\log P_\theta(x^n))}{\partial \theta^2}|_{\hat{\theta}} \right\} \cdot \delta + O(n \cdot \delta^3), \end{array}$$

where δ^T denotes a transpose of δ. The second term equals 0 because $\hat{\theta}$ is the MLE estimate, and we ignore the fourth term. Furthermore, the third term can be written as

$$\frac{1}{2} \cdot \log e \cdot n \cdot \delta^T \cdot \left\{ \frac{\partial^2(-\frac{1}{n} \cdot \ln P_\theta(x^n))}{\partial \theta^2}|_{\hat{\theta}} \right\} \cdot \delta,$$

where 'ln' denotes the natural logarithm (recall that 'log' denotes the logarithm to the base 2). Under certain suitable conditions, when $n \to \infty$, $\left\{ \frac{\partial^2(-\frac{1}{n} \ln P_\theta(x^n))}{\partial \theta^2}|_{\hat{\theta}} \right\}$ can be approximated as a k-dimensional matrix of constants $I(\theta)$ known as the 'Fisher information matrix.'

Let us next consider

$$\min_\delta \left(\log \frac{V}{\delta_1 \cdots \delta_k} - \log P_{\hat{\theta}}(x^n) + \frac{1}{2} \cdot \log e \cdot n \cdot \delta^T \cdot I(\theta) \cdot \delta\right).$$

Differentiating this formula with each δ_i and having the results equal 0, we obtain the following equations:

$$(n \cdot I(\theta) \cdot \delta)_i - \frac{1}{\delta_i} = 0, \quad (i = 1, \cdots, k). \tag{A.2}$$

Suppose that the eigenvalues of $I(\theta)$ are $\lambda_1.\cdots,\lambda_k$, and the eigenvectors are $(u_1,\cdots,u_k)$. If we consider only the case in which the axes of a cell (k-dimensional rectangular solid) in the discretized vector space are in parallel with $(u_1,\cdots,u_k)$, then (A.2) becomes

$$n\cdot\begin{pmatrix}\lambda_1 & & 0\\ & \ddots & \\ 0 & & \lambda_k\end{pmatrix}\begin{pmatrix}\delta_1\\ \vdots\\ \delta_k\end{pmatrix}=\begin{pmatrix}\frac{1}{\delta_1}\\ \vdots\\ \frac{1}{\delta_k}\end{pmatrix}.$$

Hence, we have

$$\delta_i=\frac{1}{\sqrt{n\cdot\lambda_i}}$$

and

$$n\cdot\delta^T\cdot I(\theta)\cdot\delta=k.$$

Moreover, since $\lambda_1\cdots\lambda_k=|I(\theta)|$ where '$|\mathbf{A}|$' stands for the determinant of $\mathbf{A}$, we have

$$\frac{1}{\delta_1\cdots\delta_k}=\sqrt{n}^k\cdot\sqrt{|I(\theta)|}.$$

Finally, (A.1) becomes

$$\begin{aligned}&\min_\delta\left(\log\tfrac{V}{\delta_1\cdots\delta_k}-\log P_{\tilde\theta}(x^n)\right)\\ &\approx\log(V\cdot\sqrt{n}^k\cdot\sqrt{|I(\theta)|})-\log P_{\hat\theta}(x^n)+\tfrac{1}{2}\cdot\log e\cdot k+O(\tfrac{1}{\sqrt{n}})\\ &=-\log P_{\hat\theta}(x^n)+\tfrac{k}{2}\cdot\log n+\tfrac{k}{2}\cdot\log e+\log V+\tfrac{1}{2}\cdot\log(|I(\theta)|)+O(\tfrac{1}{\sqrt{n}})\\ &=-\log P_{\hat\theta}(x^n)+\tfrac{k}{2}\cdot\log n+O(1).\end{aligned}$$

A.2 Learning a Soft Case Slot Model

I describe here a method of learning the soft case slot model defined in (3.2).

We can first adopt an existing soft clustering of words and estimate the word probability distribution $P(n|C)$ within each class by employing a heuristic method (cf., (Li and Yamanishi, 1997)). We can next estimate the coefficients (parameters) $P(C|v,r)$ in the finite mixture model.

There are two common methods for statistical estimation, Maximum Likelihood Estimation and Bayes Estimation. In their implementation for estimating the above coefficients, however, both of them suffer from computational intractability. The EM algorithm (Dempster, Laird, and Rubin, 1977) can be used to approximate the maximum likelihood estimates of the coefficients. The Markov Chain Monte-Carlo technique (Hastings, 1970; Geman and Geman, 1984; Tanner and Wong, 1987; Gelfand and Smith, 1990) can be used to approximate the Bayes estimates of the coefficients.

We consider here the use of an extended version of the EM algorithm (Helmbold et al., 1995). For the sake of notational simplicity, for some fixed v and r, let us write

$P(C|v,r), C \in \Gamma$ as $\theta_i (i = 1, \cdots, m)$ and $P(n|C)$ as $P_i(n)$. Then the soft case slot model in (3.2) may be written as

$$P(n|v,r) = \sum_{i=1}^{m} \theta_i \cdot P_i(n).$$

Letting $\theta = (\theta_1, \cdots, \theta_m)$, for a given training sequence $n_1 \cdots n_S$, the maximum likelihood estimate of θ, denoted as $\hat{\theta}$, is defined as the value that maximizes the following log likelihood function

$$L(\theta) = \frac{1}{S} \sum_{t=1}^{S} \log \left(\sum_{i=1}^{m} \theta_i \cdot P_i(n_t) \right).$$

The EM algorithm first arbitrarily sets the initial value of θ, which we denote as $\theta^{(0)}$, and then successively calculates the values of θ on the basis of its most recent values. Let s be a predetermined number. At the lth iteration ($l = 1, \cdots, s$), we calculate $\theta^{(l)} = (\theta_1^{(l)}, \cdots, \theta_m^{(l)})$ by

$$\theta_i^{(l)} = \theta_i^{(l-1)} \left(\eta (\bigtriangledown L(\theta^{(l-1)})_i - 1) + 1 \right),$$

where $\eta > 0$ (when $\eta = 1$, Helmbold et al's version simply becomes the standard EM algorithm), and $\bigtriangledown L(\theta)$ denotes

$$\bigtriangledown L(\theta) = \left(\frac{\partial L}{\partial \theta_1} \cdots \frac{\partial L}{\partial \theta_m} \right).$$

After s numbers of calculations, the EM algorithm outputs $\theta^{(s)} = (\theta_1^{(s)}, \cdots, \theta_m^{(s)})$ as an approximate of $\hat{\theta}$. It is theoretically guaranteed that the EM algorithm converges to a *local* maximum of the likelihood function (Dempster, Laird, and Rubin, 1977).

A.3 Number of Tree Cuts

If we write $F(i)$ for the number of tree cuts in a complete b-ary tree of depth i, we can show by mathematical induction that the number of tree cuts in a complete b-ary tree of depth d satisfies

$$F(d) = \Theta(2^{b^{d-1}}),$$

since clearly

$$F(1) = 1 + 1$$

and

$$F(i) = (F(i-1))^b + 1 \quad (i = 2, \cdots, d).$$

Since the number of leaf nodes N in a complete b-ary tree equals b^d, we conclude that the number of tree cuts in a complete b-ary tree is of order $\Theta(2^{\frac{N}{b}})$.

Note that the number of tree cuts in a tree depends on the structure of that tree. If a tree is what I call a 'one-way branching b-ary tree,' then it is easy to verify that the number of tree cuts in that tree is only of order $\Theta(\frac{N-1}{b-1})$. Figure A.1 shows an example one-way branching b-ary tree. Note that a thesaurus tree is an unordered tree (for the definition of an unordered tree, see, for example, (Knuth, 1973)).

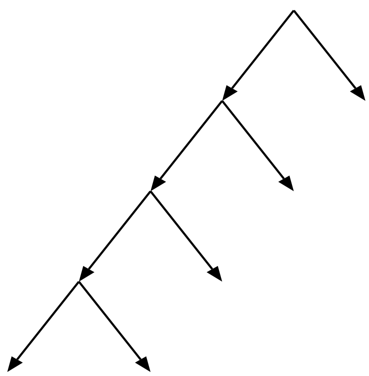


Figure A.1: An example one-way branching binary tree.

A.4 Proof of Proposition 1

For an arbitrary subtree T' of a thesaurus tree T and an arbitrary tree cut model $M = (\Gamma, \theta)$ in T, let $M_{T'} = (\Gamma_{T'}, \theta_{T'})$ denote the submodel of M that is contained in T'. Also, for any sample S, let $S_{T'}$ denote the subsample of S contained in T'. (Note that $M_T = M$, $S_T = S$.) Then, in general for any submodel $M_{T'}$ and subsample $S_{T'}$, define $L(S_{T'}|\Gamma_{T'}, \hat{\theta}_{T'})$ to be the data description length of subsample $S_{T'}$ using submodel $M_{T'}$, define $L(\hat{\theta}_{T'}|\Gamma_{T'})$ to be the parameter description length for the submodel $M_{T'}$, and define $L'(M_{T'}, S_{T'})$ to be $L(S_{T'}|\Gamma_{T'}, \hat{\theta}_{T'}) + L(\hat{\theta}_{T'}|\Gamma_{T'})$.

First for any (sub)tree T, for any (sub)model $M_T = (\Gamma_T, \hat{\theta}_T)$ which is contained in T except the (sub)model consisting only of the root node of T, and for any (sub)sample S_T contained in T, we have

$$L(S_T|\Gamma_T, \hat{\theta}_T) = \sum_{i=1}^{k} L(S_{T_i}|\Gamma_{T_i}, \hat{\theta}_{T_i}), \tag{A.3}$$

where $T_i, (i = 1, \cdots, k)$ denote the child subtrees of T.

For any (sub)tree T, for any (sub)model $M_T = (\Gamma_T, \hat{\theta}_T)$ which is contained in T

except the (sub)model consisting only of the root node of T, we have

$$L(\hat{\theta}_T|\Gamma_T) = \sum_{i=1}^{k} L(\hat{\theta}_{T_i}|\Gamma_{T_i}), \tag{A.4}$$

where $T_i, (i = 1, \cdots, k)$ denote the child subtrees of T.

When T is the entire thesaurus tree, the parameter description length for a tree cut model in T should be

$$L(\hat{\theta}_T|\Gamma_T) = \sum_{i=1}^{k} L(\hat{\theta}_{T_i}|\Gamma_{T_i}) - \frac{\log|S|}{2}, \tag{A.5}$$

where $|S|$ is the size of the entire sample. Since the second term $-\frac{\log|S|}{2}$ in (A.5) is common to each model in the entire thesaurus tree, it is irrelevant for the purpose of finding a model with the minimum description length.

We will thus use identity (A.4) both when T is a proper subtree and when it is the entire tree. (This allows us to use the same recursive algorithm (Find-MDL) in all cases.)

It follows from (A.3) and (A.4) that the minimization of description length can be done essentially independently for each subtree. Namely, if we let $L'_{min}(M_T, S_T)$ denote the minimum description length (as defined by (A.3) and (A.4)) achievable for (sub)model M_T on (sub)sample S_T contained in (sub)tree T, $\hat{P}(\eta)$ the MLE estimate of the probability for node η, and root(T) the root node of T, then we have

$$L'_{min}(M_T, S_T) = \min\{\textstyle\sum_{i=1}^{k} L'_{min}(M_{T_i}, S_{T_i}), L'(([\mathrm{root}(T)], [\hat{P}(\mathrm{root}(T))]), S_T)\}. \tag{A.6}$$

Here, $T_i, (i = 1, \cdots, k)$ denote the child subtrees of T.

The rest of the proof proceeds by induction. First, if T is a subtree having a single node, then there is only one submodel in T, and it is clearly the submodel with the minimum description length. Next, inductively assume that Find-MDL(T') correctly outputs a submodel with the minimum description length for any subtree T' of size less than n. Then, given a (sub)tree T of size n whose root node has at least two child subtrees, say $T_i : i = 1, \cdots, k$, for each T_i, Find-MDL(T_i) returns a submodel with the minimum description length by inductive hypothesis. Then, since (A.6) holds, in whichever way the if-clause on lines 8, 9 of Find-MDL is evaluated, what is returned on line 11 or line 13 will still be a (sub)model with the minimum description length, completing the inductive step.

It is easy to see that the time complexity of the algorithm is linear in the number of leaf nodes of the input thesaurus tree.

A.5 Equivalent Dependency Tree Models

We prove here that the dependency tree models based on a labeled free tree are equivalent to one another. Here, a labeled free tree means a tree in which each node is associated with one unique label and in which any node can be the root.

Suppose that the free tree we have is now rooted at X_0 (Figure A.2). The dependency tree model based on this rooted tree will then be uniquely determined. Suppose that we randomly select one other node X_i from this tree. If we reverse the directions of the links from X_0 to X_i, we will obtain another tree rooted at X_i. Another dependency tree model based on this tree will also be determined. It is not difficult to see that these two distributions are equivalent to one another, since

$$\begin{aligned} & P(X_0) \cdot P(X_1|X_0) \cdots P(X_i|X_{i-1}) \\ = & P(X_0|X_1) \cdot P(X_1) \cdots P(X_i|X_{i-1}) \\ = & \cdots \\ = & P(X_0|X_1) \cdots P(X_{i-1}|X_i) \cdot P(X_i). \end{aligned}$$

Thus we complete the proof.

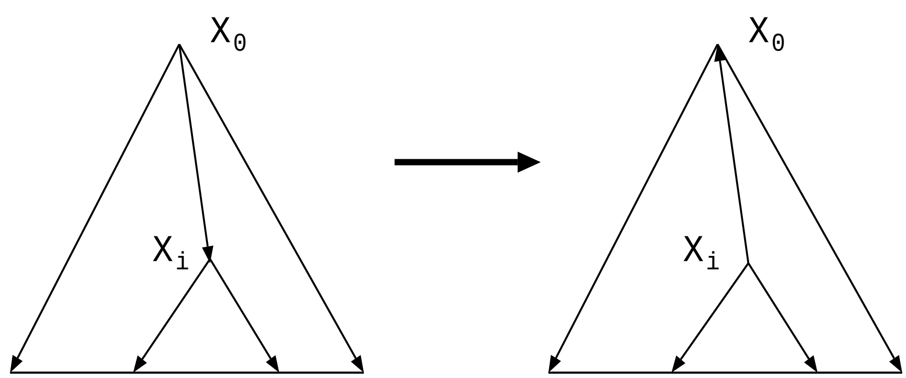


Figure A.2: Equivalent dependency tree models.

A.6 Proof of Proposition 2

We can represent any dependency forest model as

$$P(X_1, \cdots, X_n) = P(X_1|X_{q(1)}) \cdots P(X_i|X_{q(i)}) \cdots P(X_n|X_{q(n)})$$
$$0 \leq q(i) \leq n, q(i) \neq i, (i = 1, \cdots, n)$$

where $X_{q(i)}$ denotes a random variable which X_i depends on. We let $P(X_i|X_0) = P(X_i)$. Note that there exists a $j, (j = 1, \cdots, n)$ for which $P(X_j|X_{q(j)}) = P(X_j|X_0) = P(X_j)$.

The sum of parameter description length and data description length for any de-

pendency forest model equals

$$\begin{aligned}
&\textstyle\sum_{i=1}^{n} \frac{k_i-1}{2} k_{q(i)} \cdot \log N - \sum_{x_1,\cdots,x_n} f(x_1,\cdots,x_n) \cdot \log\Big(\hat{P}(x_1|x_{q(1)}) \cdots \hat{P}(x_i|x_{q(i)}) \cdots \hat{P}(x_n|x_{q(n)})\Big) \\
&= \textstyle\sum_{i=1}^{n} \frac{k_i-1}{2} k_{q(i)} \cdot \log N - \sum_{i=1}^{n} \sum_{x_i,x_{q(i)}} f(x_i,x_{q(i)}) \cdot \log \hat{P}(x_i|x_{q(i)}) \\
&= \textstyle\sum_{i=1}^{n} \Big(\frac{k_i-1}{2} k_{q(i)} \cdot \log N - \sum_{x_i,x_{q(i)}} f(x_i,x_{q(i)}) \cdot \log \hat{P}(x_i|x_{q(i)}) \Big),
\end{aligned}$$

where N denotes data size, x_i the possible values of X_i, and k_i the number of possible values of X_i. We let $k_0 = 1$ and $f(x_i, x_0) = f(x_i)$.

Furthermore, the sum of parameter description length and data description length for the independent model (i.e., $P(X_1, \cdots, X_n) = \prod_{i=1}^{n} P(X_i)$) equals

$$\begin{aligned}
&\textstyle\sum_{i=1}^{n} \frac{k_i-1}{2} \cdot \log N - \sum_{x_1,\cdots,x_n} f(x_1,\cdots,x_n) \cdot \log\Big(\prod_{i=1}^{n} \hat{P}(x_i)\Big) \\
&= \textstyle\sum_{i=1}^{n} \frac{k_i-1}{2} \cdot \log N - \sum_{i=1}^{n} \sum_{x_i} f(x_i) \cdot \log \hat{P}(x_i) \\
&= \textstyle\sum_{i=1}^{n} \Big(\frac{k_i-1}{2} \cdot \log N - \sum_{x_i} f(x_i) \cdot \log \hat{P}(x_i) \Big).
\end{aligned}$$

Thus, the difference between the description length of the independent model and the description length of any dependency forest model becomes

$$\begin{aligned}
&\textstyle\sum_{i=1}^{n} \Big(\sum_{x_i,x_{q(i)}} f(x_i,x_{q(i)}) \cdot (\log \hat{P}(x_i|x_{q(i)}) - \log \hat{P}(x_i)) - \frac{(k_i-1)\cdot(k_{q(i)}-1)}{2} \cdot \log N \Big) \\
&= \textstyle\sum_{i=1}^{n} \Big(N \cdot \hat{I}(X_i, X_{q(i)}) - \frac{(k_i-1)\cdot(k_{q(i)}-1)}{2} \cdot \log N \Big) \\
&= \textstyle\sum_{i=1}^{n} N \cdot \theta(X_i, X_{q(i)}).
\end{aligned}$$

Any dependency forest model for which there exists an i satisfying $\theta(X_i, X_{q(i)}) < 0$ is not favorable from the viewpoint of MDL because the model for which the corresponding i satisfying $\theta(X_i, X_{q(i)}) = 0$ always exists and is clearly more favorable.

We thus need only select the MDL model from those models for which for any i, $\theta(X_i, X_{q(i)}) \geq 0$ is satisfied. Obviously, the model for which $\sum_{i=1}^{n} \theta(X_i, X_{q(i)})$ is maximized is the best model in terms of MDL. What Suzuki's algorithm outputs is exactly this model, and this completes the proof.

Publication List

Reviewed Journal Papers

1. Li, H.: A Probabilistic Disambiguation Method based on Psycholinguistic Principles, (in Japanese) *Computer Software*, Vol.13, No. 6, (1996) pp. 53–65.

2. Li, H. and Abe, N.: Clustering Words with the MDL Principle, *Journal of Natural Language Processing*, Vol.4, No. 2, (1997), pp. 71–88.

3. Li, H. and Abe, N.: Generalizing Case Frames Using a Thesaurus and the MDL Principle, *Computational Linguistics*, Vol.24, No.2 (1998), pp. 217-244.

Reviewed Conference Papers

1. Li, H. and Abe, N.: Generalizing Case Frames Using a Thesaurus and the MDL Principle, *Proceedings of Recent Advances in Natural Language Processing*, (1995), pp. 239–248.

2. Abe, N. and Li, H.: On-line Learning of Binary Lexical Relations Using Two-dimensional Weighted Majority Algorithms, *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, (1995), pp. 71–88.

3. Li, H.: A Probabilistic Disambiguation Method based on Psycholinguistic Principles, *Proceedings of the 4th Workshop on Very Large Corpora*, (1996), pp. 141–154.

4. Li, H. and Abe, N.: Clustering Words with the MDL Principle, *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, (1996), pp. 4–9.

5. Li, H. and Abe, N.: Learning Dependencies between Case Frame Slots, *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, (1996), pp. 10–15.

6. Abe, N. and Li, H.:Learning Word Association Norms Using Tree Cut Pair Models , *Proceedings of the 13th International Conference on Machine Learning (ICML'96)*, (1996), pp. 71–88.

7. Li, H. and Yamanishi, K.: Document Classification Using a Finite Mixture Model, *Proceedings of the 35th Annual Meeting of Association for Computational Linguistics (ACL/EACL'97)* , (1997).

8. Li, H. and Abe, N.: Word Clustering and Disambiguation Based on Co-occurrence Data, *Proceedings of the 18th International Conference on Computational Linguistics and the 36th Annual Meeting of Association for Computational Linguistics (COLING-ACL'98)*, (1998), to appear.